Robust Interactive Simulation of Deformable Solids with Detailed Geometry using Corotational FEM

Doctoral Thesis

by

Oscar Civit Flores

Advisor: Toni Susín

PhD Programme in Computing

Barcelona, December 2015

Abstract

This thesis focuses on the interactive simulation of highly detailed deformable solids modelled with the Corotational Finite Element Method.

Starting from continuum mechanics we derive the discrete equations of motion and present a simulation scheme with support for user-in-the-loop interaction, geometric constraints and contact treatment. The interplay between accuracy and computational cost is discussed in depth, and practical approximations are analyzed with an emphasis on robustness and efficiency, as required by interactive simulation.

The first part of the thesis focuses on deformable material discretization using the Finite Element Method with simplex elements and a corotational linear constitutive model, and presents our contributions to the solution of widely reported robustness problems in case of large stretch deformations and finite element degeneration. First, we introduce a stress differential approximation for quasi-implicit corotational linear FEM that improves its results for large deformations and closely matches the fully-implicit solution with minor computational overhead. Next, we address the problem of robustness and realism in simulations involving element degeneration, and show that existing methods have previously unreported flaws that seriously threaten robustness and physical plausibility in interactive applications. We propose a new continuous-time approach, degeneration-aware polar decomposition, that avoids such flaws and yields robust degeneration recovery.

In the second part we focus on geometry representation and contact determination for deformable solids with highly detailed surfaces. Given a high resolution closed surface mesh we automatically build a coarse embedding tetrahedralization and a partitioned representation of the collision geometry in a preprocess. During simulation, our proposed contact determination algorithm finds all intersecting pairs of deformed triangles using a memory-efficient barycentric bounding volume hierarchy, connects them into potentially disjoint intersection curves and performs a topological flood process on the exact intersection surfaces to discover a minimal set of contact points. A novel contact normal definition is used to find contact point correspondences suitable for contact treatment.

From a software engineering perspective, the presented contributions are independent building blocks that can replace or complement existing techniques in an interactive simulation application. The simulation scheme and all involved numerical methods and computational geometry functionality were implemented from scratch as a set of loosely-coupled C++11 libraries.

Resum

Aquesta tesi tracta sobre la simulació interactiva de sòlids deformables amb superfícies detallades, modelats amb el Mètode dels Elements Finits (FEM) Corotacionals.

A partir de la mecànica del continuu derivem les equacions del moviment discretes i presentem un esquema de simulació amb suport per a interacció d'usuari, restriccions geomètriques i tractament de contactes. Aprofundim en la interrelació entre precisió i cost de computació, i analitzem aproximacions pràctiques fent èmfasi en la robustesa i l'eficiència necessàries per a la simulació interactiva.

La primera part de la tesi es centra en la discretització del material deformable mitjançant el Mètode dels Elements Finits amb elements de tipus símplex i un model constituent basat en elasticitat linial corotacional, i presenta les nostres contribucions a la solució de problemes de robustesa àmpliament coneguts que apareixen en cas de sobreelongament i degeneració dels elements finits. Primer introduïm una aproximació dels diferencials d'stress per a FEM linial corotacional amb integració quasi-implícita que en millora els resultats per a deformacions grans i s'apropa a la solució implícita amb un baix cost computacional. A continuació tractem el problema de la robustesa i el realisme en simulacions que inclouen degeneració d'elements finits, i mostrem que els mètodes existents presenten inconvenients que posen en perill la robustesa i plausibilitat de la simulació en aplicacions interactives. Proposem un enfocament nou basat en temps continuu, la descomposició polar amb coneixement de degeneració, que evita els inconvenients esmentats i permet corregir la degeneració de forma robusta.

A la segona part de la tesi ens centrem en la representació de geometria i la determinació de contactes per a sòlids deformables amb superfícies detallades. A partir d'una malla de superfície tancada construïm una tetraedralització englobant de forma automàtica en un preprocés, i particionem la geometria de colisió. Proposem un algorisme de detecció de contactes que troba tots els parells de triangles deformats que intersecten mitjançant una jerarquia de volums englobants en coordenades baricèntriques, els connecta en corbes d'intersecció potencialment disjuntes i realitza un procés d'inundació topològica sobre les superfícies d'intersecció exactes per tal de descobrir un conjunt mínim de punts de contacte. Usem una definició nova de la normal de contacte per tal de calcular correspondències entre punts de contacte útils per al seu tractament.

Les contribucions presentades són blocs independents que poden reemplaçar o complementar les tècniques existents en una aplicació de simulació interactiva. Hem implementat des de zero l'esquema general de simulació i tots els mètodes numèrics i de geometria computacional requerits en un conjunt de llibreries en C++11.

Acknowledgements

It's been a long and bumpy ride since 2003, when I enrolled the PhD program at UPF, and later moved to UPC-BarcelonaTech in 2007, while simultaneously working in the videogames industry at Digital Legends Entertainment (DLE).

For the best part of this endeavour Toni Susín has been my Advisor, to whom I'm deeply grateful for giving me the research and schedule freedom I needed while offering unconditional academic, organizational and moral support.

I'm also grateful to Àlvar Vinacua, my former Tutor, for thoroughly reading my manuscripts and listening with interest to crazy my ideas, and with Oliver Sander for helping me to get started with the Finite Element Method and sharing his much needed theoretical perspective on the first part of the thesis. The inspiring discussions with Unai Landa and Jordi Rovira helped me bet on the winning horse when working on contact determination for highly detailed deformable solids.

Finishing a thesis while working at DLE would not have been possible without the flexibility and support of its CEO Xavier Carrillo and its CTO Unai Landa. My gratitude extends to the whole DLE crew, and individually to Carlos, Sergi(s), Raúl and Jon, for their interest in my research and their logistic support when needed.

Also thanks to Jordi Radev for taking over a good part of my course bureaucracy at ENTI in order to help me focus on writing the thesis.

Thanks to Jordi Rovira and Vicenç Gómez for their friendship and for trying, and failing, to conquer the world with me. I'm also grateful to my friends Laura and Olga, whose encouragement has been constant, and to my "decadent" friends: Carlos, Yolanda, Manu and Eva, for always being supportive and interested in my research affairs, and sharing good advice, food and wine in similar proportions.

Huge thanks to my parents, Carme i Josep, for believing in me more than myself, for teaching me how to work hard and enjoy life at the same time, and for stoically listening to my incomprehensible technical babble.

Finally, the biggest thanks and all my love to Esther for being my partner in life and supporting me for the last 10 years, specially during the soul-crushing process of finishing this thesis, which would not exist without her.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Computational simulation of physical systems has received major interest since the early ages of computers, quickly becoming an unvaluable tool in fields such as mechanical engineering, wheather prediction, or materials science. Simulation algorithms initially stressed the available processing power and required a huge amount of time to complete, limiting user involvement in the simulation execution to the definition of initial conditions and physical parameters. As the programmability and processing power of computers increased, it became possible to execute simple simulations fast enough to correlate the dynamics' time-scale to the human perception-reaction time-scale. This fact opened the door to *interactivity*, as it allowed the user to observe the system dynamics and simultaneoulsy generate unpredictable inputs that would affect its evolution in *real-time*.

Interactive simulation has enabled the development of important application areas such as Virtual Prototyping, Training and Surgery, and during the last decade has gained immense popularity in commercial Videogames. On an academic level, research in videogame-oriented interactive simulation has become an important part of the field, stable cross-pollination with offline physically-based animation has emerged, and main conferences and journals in Computer Graphics regularly publish videogame-oriented simulation research. On an applied level, modern videogames bring complex interactive simulations to the mass-market, and consumer-level CPUs and GPUs offer enormous computational power for physics simulation, previously reserved to dedicated HPC installations.

In this thesis we focus on the interactive simulation of elastically deformable solids

for application areas that have strict robustness and efficiency requirements, but weak physical accuracy and real-world validation demands. While our main target application are videogames, other non-critical areas such as virtual entertainment and education share similar requirements.

Interactive simulation of deformable solids presents additional challenges and risks when compared to offline simulation. Ensuring robustness under unpredictable user inputs requires stable, self-correcting numerical methods that, in addition, must converge within a strictly limited computation time to meet real-time update rates. Moreover, contact determination for freely moving deformable solids must efficiently deal with dynamically changing and non-convex geometry that can result in highly complex contact configurations. As a result, unconditionally robust and efficient simulation of deformable solids can be considered an unsolved problem.

1.1 Interactive simulation in videogames

Physics simulation finds application in two different aspects of videogames: *physics-based secondary animation* and *physics-based gameplay*. The former is often used to improve visual realism by adding naturally moving details such as character hair and clothes, tree foliage, smoke, etc, that have no further effect in the game. The latter represents gameplay mechanics that use physics simulation at their core, such as vehicle driving, throwing objects or solving mechanical puzzles. Both aspects usually coexist, but have different requirements.

Secondary animation allows for aggresive simplifications and its detail can be scaled according to available computational power, even disabled, without affecting the game mechanics. Robustness requirements are high, but minor issues can be often ignored, such as character hair and clothes intersecting its body. On the other hand, physically-based gameplay mechanics greatly amplifies the importance of robustness and efficiency, as any instability may be fatal and render the game unplayable. Additionally, for multi-platform games the computational cost of physics-based gameplay must remain low enough to allow identical mechanics across the whole range of supported platforms.

In this context the main requirements of interactive simulation are, in decreasing priority

Interaction: The simulation needs to accept non-trivial external inputs at any time. Objects can be created, destroyed and modified during simulation, and external forces, constraints and contacts can transiently change the equations of motion.

Robustness: Simulation algorithms need to be numerically stable under unexpected user input and inconsistent configurations.

Efficiency: The computational cost needs to meet strict limits under all circumstances to ensure real-time interaction feedback.

Realism: Simulation results must remain realistic and avoid unnatural behaviours such as jitter. Approximations are acceptable, but must not compromise physical plausibility.

These strict requirements set an implicit limit on the complexity of the systems that can be interactively simulated. As a result, games generally use the simplest physical model possible in order to spend the existing computational power in the most effective way. At the same time, the demand for visual quality constantly increases and drives games to render the most detailed geometry possible. These divergent trends often lead to inconsistency, as visually realistic objects exhibit simplistic dynamics and collision detection that do not match visual fidelity.

In practice, gameplay physics is generally limited to articulated rigid bodies with primitive or convex geometry, for which robust and efficient simulation algorithms exist [BET14]. This results in simplistic depiction of non-rigid solid objects, such as metal structures, plastic objects, plants and trees, animals, etc, that should deform under physical interaction with other entities, but instead are generally simulated as articulated rigid bodies, and rendered as rigid or linear-blend skinned geometry.

Secondary animation can afford more complex physical models such as deformable curves or surfaces for character hair, clothes and environment detail [BMO+14], but is generally limited to small-mass objects that only interact passively with gameplay entities.

1.2 Motivation and objectives

Despite a large body of research available on the subject [NMK+06], and the existence of specific middleware [PO09], the simulation of deformable solids for gameplay purposes has not found widespread adoption due to the following reasons:

- Simulation cost: Deformable solids involve a large number of degrees of freedom. While particles and rigids have a constant number of degrees of freedom (DOF) $\mathcal{O}(1)$, the number of DOF for deformable solids at a given geometric resolution N grows with $\mathcal{O}(N^3)$, which directly translates into computational cost.
- Contact determination cost: Intersection and proximity tests between deformable non-convex geometry are expensive when compared to tests involving primitive shapes, convex polyhedrons or even non-convex static geometry.
- Visualization cost: Rendering deformable geometry is significantly more expensive than static or rigid geometry.

- Robustness: Deformable solids under arbitrary interaction can suffer from local volume collapse, inversion and self intersection, that need to be corrected to avoid physical and numerical instability.

- Control: Gameplay physics often requires the ability to *control* the dynamics of physically-based entities to fulfill game objectives, which becomes progressively more difficult and expensive as the number of DOF increases.

- Authoring: The cost of authoring an in-game object increases with its detail, and usually involves separate visualization, simulation and collision representations, as well as non-trivial configuration of physical parameters that require specific knowledge about the simulation scheme used.

While these issues represent a real challenge, we believe that with additional research and implementation efforts they could be overcome, and robust, cost-effective deformable solid simulation could substitute many current uses of articulated rigid bodies with skinning in order to improve interaction realism. With the popularization of Virtual Reality, physically based interaction in first person will become increasingly important, and emphasize the need for detailed interaction with virtual objects. Moreover, we believe that mathematical models derived from continuum mechanics, such as the Finite Element Method, are best suited for long-term success than ad-hoc models such as mass and spring networks or shape-matching solids, as they converge to the exact solution, are less dependent on material discretization and timestep, and use experimentally observable physical parameters that can be configured consistently.

Our specific research topics and contributions stem from our attempt to solve the issues encountered during the implementation of a state-of-the-art corotational linear FEM simulator, namely:

- Robustness in presence of degenerate or very stretched finite elements.

- Robustness and efficiency of contact determination and response for intersecting highly detailed embedded geometry.

1.3 Contributions

- A new approximation for corotational linear FEM stress differentials that improves robustness and energy conservation with small computational overhead (Section 3.3.4).

- A continuous-time method to correct finite element degeneracy that improves robustness and realism (Section 4.4).

- A dedicated surface representation that accurately matches visual geometry detail and accelerates contact determination (Section 5.4).
- A memory-efficient BVH structure defined in barycentric coordinates, the b-DOP-Tree, that reduces update cost under deformation (Section 5.4.3).
- A flood-based contact point generation algorithm that computes a reduced set of contact points efficiently (Section 5.5).
- A novel definition of contact normals that is robust in presence of high frequency surface detail and severe interpenetrations (Section 5.5.4.1).

1.4 Publications

The analysis of degenerate element treatment in corotational FEM and the improved Project/Reflect methods for quasi-implicit integration were published as an article in *Computer Graphics Forum*, and presented in an invited talk at *Symposium on Geometry Processing 2015*

- O. Civit-Flores, A. Susín. *Robust Treatment of Degenerate Elements in Interactive Corotational FEM Simulations.* Computer Graphics Forum, Vol 33 (6), pp 298-309 (2014), DOI:10.1111/cgf.1235

The hybrid stress differentials approximation in Section 3.3.4 and a preliminary version of degeneration-aware polar decomposition were presented at the Posters Session of the *Symposium on Computer Animation 2014*.

The contact determination algorithm for highly detailed geometry embedded in tetrahedral meshes has been accepted as a long paper at *Motion in Games 2015*

- O. Civit-Flores, A. Susín. *Fast Contact Determination for Intersecting Deformable Solids.*

CHAPTER 2

Deformable Solids

This chapter offers a brief introduction to the mathematical tools required to describe the dynamics of deformable solid objects, from continuum mechanics to material and time discretization, and introduces the specific finite element formulation with simplex elements that will be used througout the thesis.

2.1 Continuum mechanics

Continuum mechanics describes the dynamics of idealized continuous matter. The macroscopic behaviour of a material domain Ω emerges from the local properties of its infinitesimal material elements $\mathrm{d}m$. Its Lagrangian equations of motion can be derived from the conservation principles of mass, momentum and energy, expressed as integrals over the material domain, and a constitutive model that describes material reaction to deformation.

We shall focus on deformable solids with constant topology, defined by a material domain Ω that describes the undeformed solid geometry. Real world solids can exhibit very complex behaviour under deformation, such as elasticity, viscosity, plasticity and fracture, depending on their structure and material composition. For simplicity we will only deal with purely elastic homogenous materials.

2.1.1 Deformation

The *deformation map* $\phi : \Omega \to \mathbb{R}^3$ relates the deformed and reference configurations $\boldsymbol{x} = \phi(\boldsymbol{X})$, where $\boldsymbol{x}$ and $\boldsymbol{X}$ are the *spatial* and *material coordinates* respectively.

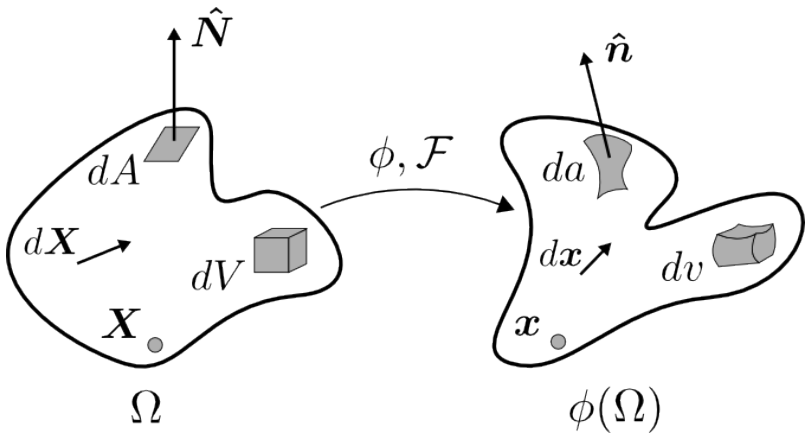

Figure 2.1: Solid domain and deformation of points, lines, areas and volumes.

The jacobian of the deformation map relative to material coordinates $\boldsymbol{X}$ is the *deformation gradient* tensor $\mathcal{F} = \frac{\partial \phi}{\partial \boldsymbol{X}} = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{X}}$, where $\mathcal{F} \in \mathbb{R}^3 \times \mathbb{R}^3$. The determinant of the deformation gradient $J = \det(\mathcal{F})$ represents the volume change fraction of a $\mathrm{d}V$. The condition $J > 0$ ensures that the deformation map is bijective. While impossible in the physical world, the degenerate cases $J = 0$ (collapse) and $J < 0$ (inversion) will need to be considered for numerical simulation.

The deformation expressions for points and differential line, area and volume elements, as shown in Figure 2.1, are

$$\boldsymbol{x} = \phi(\boldsymbol{X}) \tag{2.1}$$

$$\mathrm{d}\boldsymbol{x} = \mathcal{F}\mathrm{d}\boldsymbol{X} \tag{2.2}$$

$$\mathrm{d}\boldsymbol{a} = \hat{\boldsymbol{n}}\mathrm{d}a = J\mathcal{F}^{-T}\hat{\boldsymbol{N}}\mathrm{d}A = J\mathcal{F}^{-T}\mathrm{d}\boldsymbol{A} \tag{2.3}$$

$$\mathrm{d}v = J\mathrm{d}V \tag{2.4}$$

where $\mathrm{d}\boldsymbol{A}, \mathrm{d}\boldsymbol{a}$ are vector area elements, defined by unit normal directions $\hat{\boldsymbol{N}}, \hat{\boldsymbol{n}}$ and scalar area magnitudes A, a.

In general $\phi(\boldsymbol{X})$ and $\mathcal{F}(\boldsymbol{X})$ are spatially-varying fields in Ω. In a dynamic system $\boldsymbol{x}(t) = \phi(\boldsymbol{X}, t)$ and $\mathcal{F}(\boldsymbol{X}, t)$ also vary in time. The deformation map in the neighbourhood of a material point $\boldsymbol{X}_*$ can be approximated by the first-order Taylor expansion

$$\phi(\boldsymbol{X}) \approx \phi(\boldsymbol{X}_*) + \left.\frac{\partial \phi}{\partial \boldsymbol{X}}\right|_{\boldsymbol{X}_*} (\boldsymbol{X} - \boldsymbol{X}_*) \tag{2.5}$$

$$= \boldsymbol{x}_* + \mathcal{F}_*(\boldsymbol{X} - \boldsymbol{X}_*) \tag{2.6}$$

$$= \mathcal{F}_*\boldsymbol{X} + (\boldsymbol{x}_* - \mathcal{F}_*\boldsymbol{X}_*) \tag{2.7}$$

$$= \mathcal{F}_*\boldsymbol{X} + \boldsymbol{b}_* \tag{2.8}$$

where $\mathcal{F}_*$ and $\boldsymbol{b}_*$ are the local deformation gradient and translation. For a locally rigid deformation $\mathcal{F}_* = \mathcal{R}$ is an orthogonal rotation matrix.

2.1.2 Strain

In continuum mechanics, the strain field is defined as the magnitude of the deformation experienced by each material point in Ω. A natural way to measure deformation is using the *stretch-ratio* λ of the deformed and undeformed lengths of a line element

$$\mathrm{d}l = \lambda \mathrm{d}L \tag{2.9}$$

The deformation of a line element can be measured using the *Cauchy-Green strain tensor* $\mathcal{C}$

$$\mathrm{d}x^2 = \mathrm{d}\boldsymbol{x}^T \mathrm{d}\boldsymbol{x} = (\mathcal{F}\mathrm{d}\boldsymbol{X})^T \mathcal{F}\mathrm{d}\boldsymbol{X} = \mathrm{d}\boldsymbol{X}^T\, \mathcal{C}\, \mathrm{d}\boldsymbol{X} \tag{2.10}$$

with

$$\mathcal{C} = \mathcal{F}^T \mathcal{F} \tag{2.11}$$

The eigenvalues of $\mathcal{C}$ are the squared *principal stretches* $\lambda_1^2, \lambda_2^2, \lambda_3^2$. Cauchy strain is $\mathcal{C} = \mathcal{I}$ in case of pure rigid body motion $\mathcal{F} = \mathcal{R}$. and only vanishes if $\mathcal{F} = 0$.

The *Green strain tensor* $\mathcal{E}$ measures deformation as an actual deviation from rigid body motion

$$\mathrm{d}\boldsymbol{x}^2 - \mathrm{d}\boldsymbol{X}^2 = \mathrm{d}\boldsymbol{X}^T\, \mathcal{C}\, \mathrm{d}\boldsymbol{X} - \mathrm{d}\boldsymbol{X}^T \mathrm{d}\boldsymbol{X} = \mathrm{d}\boldsymbol{X}^T (\mathcal{C} - \mathcal{I}) \mathrm{d}\boldsymbol{X} = \mathrm{d}\boldsymbol{X}^T\, 2\mathcal{E}\, \mathrm{d}\boldsymbol{X} \tag{2.12}$$

with

$$\mathcal{E} = \frac{1}{2}(\mathcal{F}^T \mathcal{F} - \mathcal{I}) \tag{2.13}$$

which vanishes for rigid body motion $\mathcal{E}(\mathcal{R}) = 0$ as $\mathcal{R}^T \mathcal{R} = \mathcal{I}$.

The Green strain tensor $\mathcal{E}(\mathcal{F})$ is nonlinear in $\mathcal{F}$, which complicates mathematical treatment and increases computational cost. Its linearization around $\mathcal{E}(\mathcal{F} = \mathcal{I})$ yields the *Cauchy infinitesimal strain tensor*

$$\epsilon = \frac{1}{2}(\mathcal{F}^T + \mathcal{F}) - \mathcal{I} \tag{2.14}$$

that is linear in $\mathcal{F}$ and vanishes for pure translations, but not for rotations $\epsilon(\mathcal{R}) \neq 0$, which is only accurate for very small deformations.

The *corotational strain tensor* ϵ_C enforces rotation invariance by factoring out the rotation in $\mathcal{F}$ while preserving the linearity of ϵ locally in the rotated reference system. Using the polar decomposition $\mathcal{F} = \mathcal{R}\mathcal{S}$ where $\mathcal{R}$ is a rotation and $\mathcal{S}$ is symmetric, we can define it as

$$\epsilon_C = \frac{1}{2}(\mathcal{S}^T + \mathcal{S}) - \mathcal{I} = \mathcal{S} - \mathcal{I} \tag{2.15}$$

Alternative factorizations of $\mathcal{F}$ will be discussed in Chapter 3.

2.1.3 Stress

The stress σ is a measure of directional force on a material point. In general, a material point in a deformed solid can be subject to forces in all directions. The *Cauchy stress tensor* $\boldsymbol{\sigma}$ defines the *traction* $\boldsymbol{t}$ affecting a material point through a surface perpendicular to the normal direction $\hat{\boldsymbol{n}}$ as

$$\boldsymbol{t} = \boldsymbol{\sigma}\hat{\boldsymbol{n}} \tag{2.16}$$

with

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{11} & \tau_{12} & \tau_{13} \\ \tau_{21} & \sigma_{22} & \tau_{23} \\ \tau_{31} & \tau_{32} & \sigma_{33} \end{bmatrix} \tag{2.17}$$

where $\boldsymbol{\sigma}^T = \boldsymbol{\sigma}$ for all practical purposes we'll consider [Wri06], due to angular momentum balance. The diagonal entries σ_{ii} are the orthogonal normal stresses, and the off-diagonal symmetric entries $\tau_{ij} = \tau_{ji}$ are the orthogonal shear stresses. The eigenvalues of $\boldsymbol{\sigma}$ are the *principal stresses* σ_i. The traction $\boldsymbol{t}$ is, in general, not aligned with $\hat{\boldsymbol{n}}$, as it includes normal and shear components. Using Equation (2.3), the force $\boldsymbol{f}$ on a differential volume element $\mathrm{d}v$ resulting from a traction $\boldsymbol{t}$ on a differential area element $\mathrm{d}a$ is

$$\boldsymbol{f}\mathrm{d}v = \boldsymbol{t}\mathrm{d}a = \boldsymbol{\sigma}\hat{\boldsymbol{n}}\mathrm{d}a = \boldsymbol{\sigma}J\mathcal{F}^{-T}\hat{\boldsymbol{N}}\mathrm{d}A = \mathcal{P}\hat{\boldsymbol{N}}\mathrm{d}A \tag{2.18}$$

where

$$\mathcal{P} = J\boldsymbol{\sigma}\mathcal{F}^{-T} \tag{2.19}$$

is the *first Piola-Kirchhoff stress tensor*, that relates material tractions to spatial tractions in a computationally efficient way [TBHF03].

Stress can be caused by external surface tractions (eg. contact), or by internal material forces (eg. elasticity), as described in the next section.

2.1.4 Elasticity

Elastic solids react to changes in shape with forces that oppose deformation and work to return each deformed volume element $\mathrm{d}v$ to its local reference configuration. In general, elasticity can be defined as an arbitrary relationship between stress and deformation $\boldsymbol{\sigma} = \mathcal{G}(\mathcal{F})$. For simplicity we will only deal with *hyperelastic materials*, for which a conservative *elastic potential energy density* $\Psi(\mathcal{F})$ can be defined as a function of the deformation $\mathcal{F}$. The forces induced by a potential energy can be computed as the energy gradient

$$\boldsymbol{f} = -\frac{\partial\Psi}{\partial\boldsymbol{x}} \tag{2.20}$$

however, for computational purposes it will be more convenient to express them through the first Piola-Kirchhoff stress tensor, that can be computed from the energy as

$$\mathcal{P}(\mathcal{F}) = \frac{\partial\Psi(\mathcal{F})}{\partial\mathcal{F}} \tag{2.21}$$

We will limit the discussion to *isotropic*, and *rotationally invariant* constitutive models. Isotropy requires energy invariance $\Psi(\mathcal{F}\mathcal{Q}) = \Psi(\mathcal{F})$ under local material rotation $\mathcal{Q}$, and applies to materials that have the same elastic response along all possible deformation directions. Rotational invariance requires the elastic energy to be constant $\Psi(\mathcal{R}\mathcal{F}) = \Psi(\mathcal{F})$ under global material rotation $\mathcal{R}$, and ensures independence from rigid body motion. Constitutive models using the Green strain tensor $\mathcal{E}(\mathcal{C})$ automatically fulfill this last property, as $\mathcal{C}(\mathcal{R}\mathcal{F}) = (\mathcal{R}\mathcal{F})^T(\mathcal{R}\mathcal{F}) = \mathcal{F}^T(\mathcal{R}^T\mathcal{R})\mathcal{F} = \mathcal{F}^T\mathcal{F}$.

Isotropic homogeneous materials can be completely specified with only two constants. For mathematical simplicity we'll use the *Lamé parameters* λ and μ. An equivalent set of parameters , *Young's modulus* E and the *Poisson ratio* ν, will be used in later chapters. The relationship between these two sets of parameters is

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}, \quad \mu = \frac{E}{2(1+\nu)} \tag{2.22}$$

We will now describe a few material models used in interactive and CG simulation literature and provide their energy and stress expressions for future reference.

2.1.4.1 Linear Material

Using the infinitesimal strain tensor ϵ we obtain a completely linear model, where forces depend linearly on displacements. However, the resulting energy is not rotationally invariant and results in non-zero forces under rigid body motion as $\epsilon(R) \neq 0$, which is only applicable to small deformations.

$$\Psi_{\mathrm{L}} = \mu\|\epsilon\|_F^2 + \frac{\lambda}{2}\operatorname{tr}^2(\epsilon) \tag{2.23}$$

$$\mathcal{P}_{\mathrm{L}} = 2\mu\epsilon + \lambda\operatorname{tr}(\epsilon)\mathcal{I} \tag{2.24}$$

2.1.4.2 Linear Corotated Material

Using the corotational strain tensor in the linear material definition yields a rotationally invariant constitutive model that preserves the mathematical and computational simplicity of the linear material but allows larger deformations.

$$\Psi_{\mathrm{C}} = \mu\|\epsilon_C\|_F^2 + \frac{\lambda}{2}\operatorname{tr}^2(\epsilon_C) \tag{2.25}$$

$$\mathcal{P}_{\mathrm{C}} = \mathcal{R}[2\mu\epsilon_C + \lambda\operatorname{tr}(\epsilon_C)\mathcal{I}] \tag{2.26}$$

2.1.4.3 Corrected Corotated Material

This nonlinear constitutive model was described in [SHST12] and uses the volume change fraction term $J = \det(\mathcal{F})$ to prevent stretching problems detected in the linear

corotated material and increases its resistance to collapse.

$$\Psi_{\mathrm{CC}} = \mu\|\epsilon_C\|_F^2 + \frac{\lambda}{2}(J-1)^2 \tag{2.27}$$

$$\mathcal{P}_{\mathrm{CC}} = \mathcal{R}[2\mu\epsilon_C + \lambda(J-1)J\mathcal{S}^{-1}] \tag{2.28}$$

2.1.4.4 St.Venant-Kirchhoff

The continuum equivalent of Hooke's law using the Green's strain tensor yields the St.Venant-Kirchhoff nonlinear material model that is rotationally invariant but becomes progressively weaker as it tends to collapse, which is highly inconvenient for interactive simulation.

$$\Psi_{\mathrm{VK}} = \mu\|\mathcal{E}\|_F^2 + \frac{\lambda}{2}\operatorname{tr}^2(\mathcal{E}) \tag{2.29}$$

$$\mathcal{P}_{\mathrm{VK}} = \mathcal{F}[2\mu\mathcal{E} + \lambda\operatorname{tr}(\mathcal{E})\mathcal{I}] \tag{2.30}$$

2.1.4.5 Neo-Hookean

The neo-Hookean material strongly opposes collapse by including energy terms that grow to infinity as the volume change fraction term $J = \det(\mathcal{F})$ approaches 0. Nonlinearity and infinite energy barrier terms complicate efficient numerical solution and limit its applicability to interactive simulation.

$$\Psi_{\mathrm{NH}} = \mu\operatorname{tr}(\mathcal{E}) - \mu\log(J) + \frac{\lambda}{2}\log^2(J) \tag{2.31}$$

$$\mathcal{P}_{\mathrm{NH}} = \mu(\mathcal{F} - \mu\mathcal{F}^{-T}) + \lambda\log(J)\mathcal{F}^{-T} \tag{2.32}$$

2.1.5 Equations of motion

The Lagrangian equations of motion for a deformable solid can be derived from the conservation of mass and momentum in material coordinates. Using Equation (2.4) mass conservation can be stated as

$$m = \int_{\phi(\Omega)} \rho\,\mathrm{d}v = \int_{\Omega} J\rho\,\mathrm{d}V = \int_{\Omega} \rho_0\,\mathrm{d}V \tag{2.33}$$

momentum conservation in spatial coordinates under conservative elastic forces can be stated as

$$\int_{\phi(\Omega)} \frac{1}{2}\rho\dot{\boldsymbol{x}}\,\mathrm{d}v + \int_{\partial\phi(\Omega)} \boldsymbol{\sigma}\hat{\boldsymbol{n}}\,\mathrm{d}a = 0 \tag{2.34}$$

where the first summand is the total momentum and using Equation (2.33) becomes

$$\int_{\phi(\Omega)} \frac{1}{2}\rho\dot{\boldsymbol{x}}\,\mathrm{d}v = \int_{\Omega} \frac{1}{2}\rho_0\dot{\boldsymbol{x}}\,\mathrm{d}V \tag{2.35}$$

and the second summand is the total elastic force in $\phi(\Omega)$, that using Equation (2.18) and the divergence theorem can be transformed into

$$\int_{\partial\phi(\Omega)} \boldsymbol{\sigma}\hat{\boldsymbol{n}}\,\mathrm{d}a = \int_{\partial\Omega} \mathcal{P}\hat{\boldsymbol{N}}\,\mathrm{d}A = \int_{\Omega} \mathrm{div}_{\boldsymbol{X}}\,\mathcal{P}\,\mathrm{d}V \tag{2.36}$$

Finally, we can rewrite angular momentum balance $\boldsymbol{\sigma}^T = \boldsymbol{\sigma}$ in terms of $\mathcal{P}$ using Equation (2.19), and obtain the balance equations

$$J\rho - \rho_0 = 0 \qquad \text{mass balance} \tag{2.37}$$

$$\rho_0\ddot{\boldsymbol{x}} - \mathrm{div}_{\boldsymbol{X}}\,\mathcal{P} = 0 \qquad \text{linear momentum balance} \tag{2.38}$$

$$\mathcal{F}\mathcal{P}^T = \mathcal{P}\mathcal{F}^T \qquad \text{angular momentum balance} \tag{2.39}$$

The equations of motion Equation (2.38) define a partial differential equation (PDE) that is first-order in material coordinates and second order in time. No closed-form solution can be expected to be found for arbitrary solid geometry and material models. Computational solution will be possible in general, and requires the discretization of material and time domains, that will be discussed individually in the following sections.

2.2 Material discretization

A deformable solid defined by a continuum material domain $\Omega \in \mathbb{R}$ has an *infinite* number of degrees of freedom (DOF). Several Lagrangian discretization methods have been proposed [NMK+06]. Mesh-based methods include Finite Differences [TPBF87], the Finite Element Method (FEM) [Wri06] and the Finite Volume Method (FVM) [EGH00]. Alternative discretizations such as meshless methods [FM04], and shape-matching [BPWG07] are also possible but will not be considered further.

Mesh-based Lagrangian discretization generally results in a set of N nodes $\{\boldsymbol{x}_i\}$ stacked in a position vector $\boldsymbol{x} = \left[\boldsymbol{x}_1^T \ldots \boldsymbol{x}_N^T\right]^T \in \mathbb{R}^{3\times N}$ that represents the DOF of the system, coupled by the discretized forces given by the constitutive material model. Mass balance will be automatically fulfilled, as each discretization element will be assigned a constant fraction of the total mass according to the volume it represents.

For a material-discretized solid with stacked node position, velocity and acceleration vectors $\boldsymbol{x}$, $\dot{\boldsymbol{x}}$, $\ddot{\boldsymbol{x}}$, the dynamics equations derived from Equation (2.38) are a set of coupled second order Ordinary Differential Equations (ODE) in time

$$\mathcal{M}\ddot{\boldsymbol{x}} = \boldsymbol{f}_s(\boldsymbol{x}) + \boldsymbol{f}_d(\boldsymbol{x}, \dot{\boldsymbol{x}}) + \boldsymbol{f}_{\text{ext}} \tag{2.40}$$

where $\mathcal{M}$ is the constant mass matrix, and $\boldsymbol{f}_s$, $\boldsymbol{f}_d$, $\boldsymbol{f}_{\text{ext}}$ are the elastic, damping and external forces, respectively. The nature of the mass matrix and the elastic and damping forces depend on the actual discretization method used. For efficiency, we'll

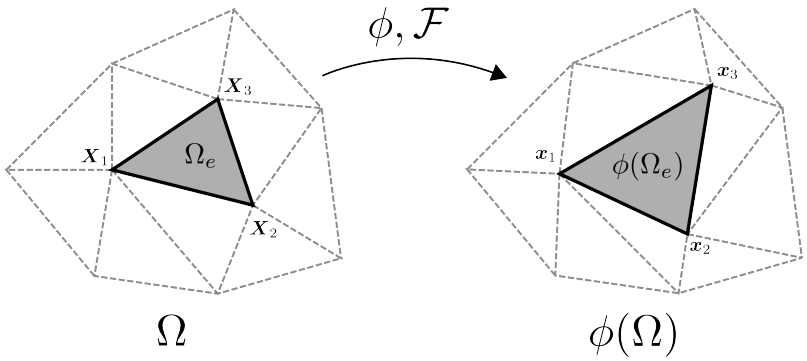

Figure 2.2: Two-dimensional material domain Ω partitioned into simplex finite elements Ω_e.

assume a diagonally *lumped mass matrix*. Elastic and damping forces can be written using the stiffness $\mathcal{K}$ and damping $\mathcal{C}$ matrices as

$$\boldsymbol{f}_s(\boldsymbol{x}) = -\mathcal{K}(\boldsymbol{x})\boldsymbol{x} \tag{2.41}$$

$$\boldsymbol{f}_d(\boldsymbol{x}, \dot{\boldsymbol{x}}) = -\mathcal{C}(\boldsymbol{x})\dot{\boldsymbol{x}} \tag{2.42}$$

For quasistatic simulations, the equilibrium equations can be derived from Equation (2.40) assuming $\dot{\boldsymbol{x}} = \ddot{\boldsymbol{x}} = 0$, and result in

$$\boldsymbol{f}_s(\boldsymbol{x}) = -\boldsymbol{f}_{\text{ext}} \tag{2.43}$$

2.2.1 FEM

In the Finite Element Method the global domain Ω is partitioned into a set of subdomains $\{\Omega_e\}$, the *finite elements*, so that $\Omega = \bigcup \Omega_e$. A set of nodes $\boldsymbol{X}_i \in \mathbb{R}^3$ determines the actual geometry of each finite element in reference configuration, and their deformed positions $\boldsymbol{x}_i \in \mathbb{R}^3$ become the dynamics degrees of freedom (Figure 2.2). The total elastic energy can be expressed as the sum of subdomain energies

$$\Pi = \int_\Omega \Psi(\mathcal{F}) = \sum_e \int_{\Omega_e} \Psi(\mathcal{F}) = \sum_e \Pi_e \tag{2.44}$$

and the elastic force $\boldsymbol{f}_s$ results from the contribution of all elements

$$\boldsymbol{f}_s = -\frac{\partial \Pi}{\partial \boldsymbol{x}} = -\sum_e \frac{\partial \Pi_e}{\partial \boldsymbol{x}} \tag{2.45}$$

In practice, only finite elements adjacent to a given node have non-zero contribution to its elastic force, and the global elastic force vector $\boldsymbol{f}_s$ can be computed by

accumulation of per-element forces $\boldsymbol{f}_e = \left[\boldsymbol{f}_1^T \dots \boldsymbol{f}_4^T\right]^T$ on its local nodes $\boldsymbol{x}_1 \dots \boldsymbol{x}_4$

$$\boldsymbol{f}_e = -\frac{\partial \Pi_e}{\partial \boldsymbol{x}_e} \tag{2.46}$$

The evaluation of Π_e and $\frac{\partial \Pi_e}{\partial \boldsymbol{x}_e}$ depends on the actual shape of the finite elements, that determines the interpolation of $\Psi(\mathcal{F})$ inside Ω_e. There is a vast catalogue of element shapes that have been used in FEM, with different accuracy and computational properties. A detailed discussion is out of the scope of this thesis (see [Fel04], for example). We will focus on simplex elements, which allow for convenient simplifications that result in a computationally efficient implementation. The interior of simplex elements can be parametrized using barycentric coordinates, and used to linearly interpolate any magnitude evaluated at the nodes. Moreover, their deformation gradient is constant, which simplifies the analytic evaluation of subdomain integrals. On the negative side, simplex elements are the least accurate, as they only offer C^0 continuity across element boundaries, and may result in *locking* for highly incompressible materials [Hau04].

2.2.1.1 Simplex interpolation

A D-dimensional simplex defined by $D+1$ vertices $\boldsymbol{v}_1 \dots \boldsymbol{v}_{D+1}$ induces a set of barycentric coordinates $\xi_1 \dots \xi_{D+1}$ that allow expressing any interior position $\boldsymbol{v}$ as a convex combination of simplex vertices

$$\boldsymbol{v} = \sum_{i=1}^{D+1} \xi_i \boldsymbol{v}_i, \quad \sum_{i=1}^{D+1} \xi_i = 1, \quad \xi_i \geq 0 \tag{2.47}$$

Using the barycentric coordinates, any magnitude $f(\boldsymbol{v})$ defined at the vertices of the simplex can be linearly interpolated as

$$f(\boldsymbol{v}) = \sum_{i=1}^{D+1} \xi_i f(\boldsymbol{v}_i) \tag{2.48}$$

The barycentric coordinates are compactly defined by

$$\begin{bmatrix} 1 \\ \boldsymbol{v} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & \dots & 1 \\ \boldsymbol{v}_1 & \dots & \boldsymbol{v}_{D+1} \end{bmatrix}}_{\mathcal{B}(\boldsymbol{v}_1,\dots,\boldsymbol{v}_{D+1})} \begin{bmatrix} \xi_1 \\ \vdots \\ \xi_D \end{bmatrix} \tag{2.49}$$

2.2.1.2 Simplex element deformation

The deformation map $\phi : \Omega \to \mathbb{R}^D$ can be approximated by piecewise affine deformations $\phi_e : \Omega_e \to \mathbb{R}^D$ with $\phi_e(\boldsymbol{r}) = \mathcal{F}\boldsymbol{r} + \boldsymbol{b}$. Assuming reference node positions

$\boldsymbol{r}_1 \dots \boldsymbol{r}_{D+1}$ and deformed node positions $\boldsymbol{x}_1 \dots \boldsymbol{x}_{D+1}$, the simplex deformation $\mathcal{G}_m^s$ is

$$\begin{bmatrix} 1 \\ \boldsymbol{x} \end{bmatrix} = \mathcal{G}_m^s \begin{bmatrix} 1 \\ \boldsymbol{r} \end{bmatrix}, \quad \mathcal{G}_m^s = \mathcal{B}_s \mathcal{B}_m^{-1} = \begin{bmatrix} 1 & 0 \\ b & \mathcal{F} \end{bmatrix}$$

where $\mathcal{B}_m = \mathcal{B}(\boldsymbol{r}_1, \dots, \boldsymbol{r}_{D+1})$ and $\mathcal{B}_s = \mathcal{B}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_{D+1})$ are the material and spatial barycentric interpolation matrices, $\mathcal{F}$ is the deformation gradient and $\boldsymbol{b}$ is the global translation vector, both *constant* over the whole simplex element. For tetrahedral elements, we have

$$\mathcal{F} = \mathcal{D}_s \mathcal{D}_m^{-1} \tag{2.50}$$

$$\mathcal{D}_m = \begin{bmatrix} \boldsymbol{r}_{12} & \boldsymbol{r}_{13} & \boldsymbol{r}_{14} \end{bmatrix} \tag{2.51}$$

$$\mathcal{D}_s = \begin{bmatrix} \boldsymbol{x}_{12} & \boldsymbol{x}_{13} & \boldsymbol{x}_{14} \end{bmatrix} \tag{2.52}$$

with reference and deformed edge vectors $\boldsymbol{r}_{ij} = \boldsymbol{r}_j - \boldsymbol{r}_i$ and $\boldsymbol{x}_{ij} = \boldsymbol{x}_j - \boldsymbol{x}_i$ respectively. The global translation $\boldsymbol{b}$ does not affect the translation invariant elastic forces and is not explicitly computed.

2.2.1.3 Elastic forces on simplex element nodes

The constant deformation gradient $\mathcal{F}$ results in a constant stress $\mathcal{P}(\mathcal{F})$ over each tetrahedral element. The elastic forces on the nodes induced by its deformation can be efficiently computed as shown in [SB12]

$$\begin{bmatrix} \boldsymbol{f}_2 & \boldsymbol{f}_3 & \boldsymbol{f}_4 \end{bmatrix} = -V_e \mathcal{P} \mathcal{D}_m^{-T} \tag{2.53}$$

$$\boldsymbol{f}_1 = -\boldsymbol{f}_2 - \boldsymbol{f}_3 - \boldsymbol{f}_4 \tag{2.54}$$

where $\mathcal{D}_m$ and $V_m = \frac{1}{6} \det(\mathcal{D}_m)$ only depend on the reference configuration and can be precomputed for efficiency, and Equation (2.54) results from momentum conservation. Per-element forces are accumulated into the global vector $\boldsymbol{f}_s$ to obtain per-node forces.

2.3 Time discretization

For interactive simulation we need to solve an initial value problem for the equations of motion Equation (2.40) subjecct to unpredictable external forces $\boldsymbol{f}_{\text{ext}}(t)$. This set of coupled second order ODE does not have an analytic solution in general, and needs to be solved numerically by discretization in the time domain. Given initial conditions $\boldsymbol{x}(0)$ and $\dot{\boldsymbol{x}}(0)$ we want to produce a sequence of configurations $\boldsymbol{x}^n$ and velocities $\dot{\boldsymbol{x}}^n$ at instants $t^n < t^{n+1}$, $t^n \in [0, \infty)$ that approximate the continuous solutions $\boldsymbol{x}(t^n)$ and $\dot{\boldsymbol{x}}(t^n)$.

The most direct approach is to recast the second order ODE into first order form using the state-variable formulation

$$\underbrace{\frac{d}{dt}\left(\begin{bmatrix}\boldsymbol{x}\\ \dot{\boldsymbol{x}}\end{bmatrix}\right)}_{\dot{\boldsymbol{y}}} = \begin{bmatrix}\dot{\boldsymbol{x}}\\ \ddot{\boldsymbol{x}}\end{bmatrix} = \underbrace{\begin{bmatrix}\dot{\boldsymbol{x}}\\ \mathcal{M}^{-1}(\boldsymbol{f}_s + \boldsymbol{f}_d + \boldsymbol{f}_{\text{ext}})\end{bmatrix}}_{f(\boldsymbol{y},t)} \tag{2.55}$$

and use a standard numerical integration method to solve the first-order ODE $\dot{\boldsymbol{y}} = f(\boldsymbol{y}, t)$. A generic integration scheme to advance time $t^n \to t^{n+1}$ during a timestep $h = t^{n+1} - t^n$ can be written as

$$\boldsymbol{y}^{n+1} = F(\boldsymbol{y}^{n+1}, \boldsymbol{y}^n, \boldsymbol{y}^{n-1}, \ldots, \boldsymbol{y}^{n-k}, t^n, h) \tag{2.56}$$

There are many families and variants of numerical integration methods that differ in the definition of F. We will only cover the methods relevant to our interactive context, see [ES04] for a more general overview. Integration schemes are classified as *explicit*, if F only requires information available at t^n, or *implicit*, if it depends on information at t^{n+1}. In addition, integration schemes are classified as *one-step* if the oldest state used is $\boldsymbol{y}^n$ or *multi-step* if they include previous states $\boldsymbol{y}^{n-1}, \ldots, \boldsymbol{y}^{n-k}$.

The most relevant aspects of a numerical integration scheme are its *accuracy* and *stability*. Accuracy determines how well the discrete dynamics approximates the continuous solution, which affects qualitative properties of motion such as energy conservation. In general, accuracy strongly depends on the timestep h, and integration schemes are classified according to their error order m for an error upper bound $\mathcal{O}(h^m)$. Stability also depends on the timestep h and determines if the sequence of approximate states $\boldsymbol{y}^n$ can diverge and become unbounded.

For interactive simulation efficiency is of foremost importance, and the main factors that determine it are the size of the timestep h and the computational cost per step. Timestep size determines how many integration steps are required to advance simulation during a given time interval $[t_0, t_1]$. The cost per simulation step depends on the cost of evaluating F, which may involve several evaluations of f, which in our case requires $\mathcal{M}^{-1}(\boldsymbol{f}_s + \boldsymbol{f}_d + \boldsymbol{f}_{\text{ext}})$. In case of implicit integration, the cost per step also includes the solution of a potentially non-linear equation on the unknown future state $\boldsymbol{y}^{n+1}$.

2.3.1 One-step methods

2.3.1.1 Explicit Euler

An explicit Euler step results from direct application of Taylor expansion at t^n and truncation of $\mathcal{O}(h^2)$ terms

$$\boldsymbol{y}^{n+1} = \boldsymbol{y}^n + hf(\boldsymbol{y}^n, t^n) \tag{2.57}$$

which for a second order ODE translates to

$$\boldsymbol{x}^{n+1} = \boldsymbol{x}^n + h\dot{\boldsymbol{x}}^n \tag{2.58}$$

$$\dot{\boldsymbol{x}}^{n+1} = \dot{\boldsymbol{x}}^n + h\ddot{\boldsymbol{x}}(\boldsymbol{x}^n, \dot{\boldsymbol{x}}^n, t^n) \tag{2.59}$$

This method requires a single force evaluation per step, but is unconditionally unstable in absence of damping for all practical problems, and requires very small timesteps even if damping is present. Accuracy is $\mathcal{O}(h)$ from the Taylor expansion approximation.

2.3.1.2 Symplectic Euler

Using updated velocities in the position update of explicit Euler results in the symplectic Euler method

$$\boldsymbol{x}^{n+1} = \boldsymbol{x}^n + h\dot{\boldsymbol{x}}^{n+1} \tag{2.60}$$

$$\dot{\boldsymbol{x}}^{n+1} = \dot{\boldsymbol{x}}^n + h\ddot{\boldsymbol{x}}(\boldsymbol{x}^n, \dot{\boldsymbol{x}}^n, t^n) \tag{2.61}$$

With essentially the same computational cost as explicit Euler due to a single force evaluation per step, this explicit method achieves significantly better stability and accuracy [Lac07], however, still requires very small steps for *stiff* problems.

2.3.1.3 Runge-Kutta

Increased accuracy and stability can be achieved with additional evaluations of f. The Runge-Kutta family offers explicit methods of order m that result in $\mathcal{O}(h^m)$ accuracy at increasing computational cost. As an example, the fourth-order RK4 method is

$$\boldsymbol{y}^{n+1} = \boldsymbol{y}^n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{2.62}$$

with (2.63)

$$k_1 = f(\boldsymbol{y}^n, t^n) \tag{2.64}$$

$$k_2 = f(\boldsymbol{y}^n + \frac{h}{2}k_1, t^n + \frac{h}{2}) \tag{2.65}$$

$$k_3 = f(\boldsymbol{y}^n + \frac{h}{2}k_2, t^n + \frac{h}{2}) \tag{2.66}$$

$$k_4 = f(\boldsymbol{y}^n + hk_3, t^n + h) \tag{2.67}$$

Despite increased accuracy, explicit Runge-Kutta higher-order methods do not improve the stable timestep upper bound significantly enough to be considered efficient, specially for stiff problems [Lac07].

2.3.1.4 Implicit Euler

An implicit Euler step requires solving

$$\boldsymbol{y}^{n+1} = \boldsymbol{y}^n + hf(\boldsymbol{y}^{n+1}, t^{n+1}) \tag{2.68}$$

that translates to

$$\boldsymbol{x}^{n+1} = \boldsymbol{x}^n + h\dot{\boldsymbol{x}}^{n+1} \tag{2.69}$$

$$\dot{\boldsymbol{x}}^{n+1} = \dot{\boldsymbol{x}}^n + h\ddot{\boldsymbol{x}}(\boldsymbol{x}^{n+1}, \dot{\boldsymbol{x}}^{n+1}, t^{n+1}) \tag{2.70}$$

The solution process involves using Newton-Raphson to find a sequence of candidate states $\boldsymbol{x}_k^{n+1}$ that is expected to converge to the exact solution. Each Newton-Raphson step requires 1 force evaluation and the construction and solution of a system of equations $\mathcal{A}\Delta\boldsymbol{x} = \boldsymbol{b}$. Implicit Euler is unconditionally stable if solved exactly, but is known to introduce numerical dissipation and results in low accuracy $\mathcal{O}(h)$. See Section 3.2 for a detailed discussion and algorithm.

2.3.2 Multi-step methods

Multi-step methods try to leverage continuity to achieve good accuracy with a low computational cost per step [ES04]. The Adams-Bashforth family of methods allows accuracy order m using $m-1$ history terms and single force evaluation per step. The Adams-Moulton and Backwards Differentiation Formulas (BDF) families offer implicit multi-step methods with improved stability.

However, in interactive simulation the continuity of trajectories $\boldsymbol{y}(t)$ and forces $\boldsymbol{f}_{\text{ext}}(t)$ cannot be guaranteed due to frequent collisions and arbitrary user interaction. Discontinuity requires *restarting* multi-step methods with consistent values for the history terms, which increases computational cost. An efficient linearized 2-step BDF method with variable timestep has been successfully applied to cloth simulation [HET01] in a non-interactive context.

2.3.3 Adaptive methods

Both accuracy and stability can greatly benefit from adaptive timestep control and prediction-correction schemes. Such methods monitor the error and adapt the computations to keep it under control [ES04]. Adaptive methods can achieve optimal efficiency, that is, guarantee a given accuracy threshold while minimizing the total computational cost of integration during a time interval $[t_0, t_1]$.

However, in interactive simulation the total simulation cost is not as important as meeting a strict real-time deadline at every single frame seen by the user, and the emphasis is on advancing a fixed time Δt, typically $1/60s$, within a nearly-constant CPU time *budget*, typically much smaller than Δt. Moreover, applying adaptive schemes in complex simulations with many different interacting objects requires additional effort to avoid forcing the most restrictive timestep on all objects. In many applications there is essentially no freedom to increase the CPU time dedicated to integration beyond a small margin, which precludes the use of sophisticated adaptive strategies.

2.3.4 Discussion

After evaluating existing numerical integration schemes and discarding both multi-step and adaptive methods for the previously detailed reasons, we're limited to one-step methods with a fixed step h. Stability is an absolute priority in interactive simulation, and explicit methods require very small h to ensure it for moderately stiff problems, a problem that remains even for higher-order methods.

Implicit integration is generally more expensive that simply evaluating a function F that yields the next state, but generally results in much better stability bounds than explicit methods, which allows for larger timesteps. Unconditionally stable methods allow for large integration steps, up to $h = \Delta t$, and are generally preferred [PO09] when robustness and nearly-constant CPU cost are a priority.

Despite its low accuracy and numerical dissipation, we will use Implicit Euler througout this thesis, as detailed in Section 3.2. Alternative methods will be considered in future work.

CHAPTER 3

Robust Interactive FEM

In this chapter we will present our interactive simulation scheme for deformable solids based in corotational linear FEM. The tradeoffs between accuracy and computational cost will be discussed, and practical approximations will be analyzed, with an emphasis on robustness and efficiency.

A stress differential approximation for quasi-implicit corotational linear FEM that improves its results for large deformations with a small computational overhead will be introduced in Section 3.3.4.

Different kinds of interactions that transiently change the free body dynamics and their practical implementation will be considered in Section 3.4.

3.1 Interactive simulation

The high-level architecture of our interactive simulation scheme is shown in Figure 3.1. In this chapter we will focus on the Time-Stepping phase, that advances simulation time by integrating the equations of motion subject to transient interaction and constraints.

In order to meet the requirements detailed in the Introduction we made the following a priori decisions, based on the analysis of previous work and our own experiments with alternative approaches:

- We choose the Finite Element Method due to its solid continuum mechanics foundation that ensures physical realism and empirically observable parameters.

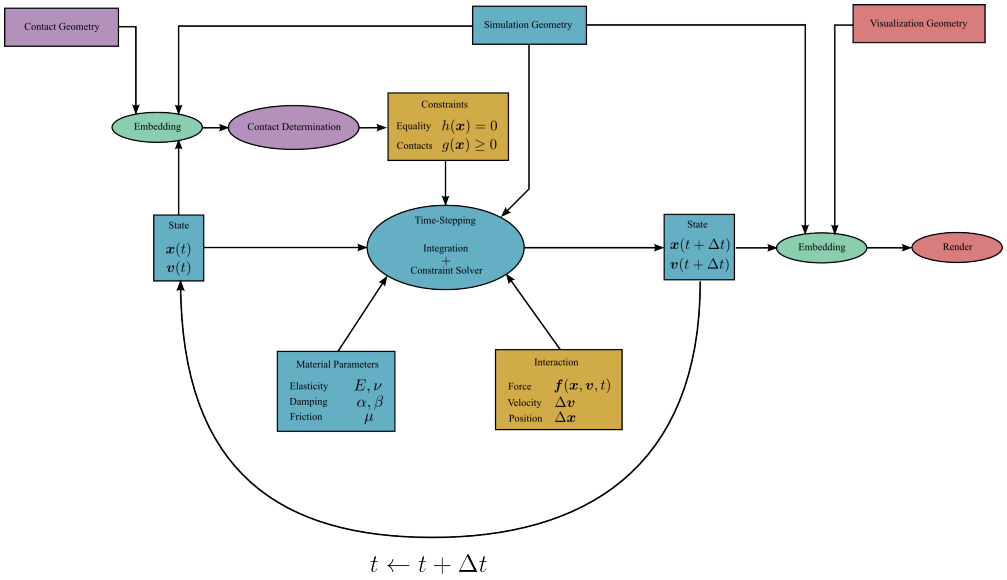

Figure 3.1: Phases of interactive simulation, with separate aspects shown in different colors: Dynamics model (blue), Transient interactions (orange), Contact (violet), Embedding (green) and Visualization (red).

- We choose simplex elements for their simplicity, that benefits both simulation and geometry embedding for visualization and contact determination.

- We choose the corotational linear constitutive model as the simplest material model that achieves sufficient realism and moderate accuracy for large deformations while remaining computationally efficient due to its near-linearity.

- We choose Implicit Euler integration for its well studied robustness and efficiency, despite its inferior energy conservation when compared to more advanced numerical integration methods.

- We use iterative linear system solvers for their convenient tradeoff between accuracy and computational cost.

- We use coarse simulation meshes with embedded detailed geometry in order to decouple the complexity of the dynamics, contact and visualization geometry representations.

- We handle contacts using local impulse-based response for simplicity, efficiency and robustness in presence of large numbers of embedded contact points.

3.2 Implicit Euler integration

An implicit Backwards-Euler integration step from n to $n+1$ with length Δt requires solving

$$\boldsymbol{x}^{n+1} = \boldsymbol{x}^n + \Delta t \boldsymbol{v}^{n+1} \tag{3.1}$$

$$\boldsymbol{v}^{n+1} = \boldsymbol{v}^n + \Delta t \mathcal{M}^{-1} \boldsymbol{f}^{n+1} \tag{3.2}$$

where Equation (3.2) is nonlinear. A solution can be found with an iterative Newton-Raphson (NR) method by linearization of $\boldsymbol{f}^{n+1}$. At each NR step k we define a position approximation $\boldsymbol{x}_k^{n+1}$ and correction $\Delta\boldsymbol{x} = \boldsymbol{x}_{k+1}^{n+1} - \boldsymbol{x}_k^{n+1}$, initialized with $\boldsymbol{x}_0^{n+1} = \boldsymbol{x}^n$, and a velocity approximation $\boldsymbol{v}_k^{n+1} = \frac{1}{\Delta t}(\boldsymbol{x}_k^{n+1} - \boldsymbol{x}^n)$, and solve the system of linear equations

$$\mathcal{A}_k^{n+1} \Delta\boldsymbol{x} = \boldsymbol{b}_k^{n+1} \tag{3.3}$$

with

$$\mathcal{A}_k^{n+1} = \mathcal{M} - \Delta t \frac{\partial \boldsymbol{f}_d}{\partial \boldsymbol{v}}\Big|_{\boldsymbol{x}_k^{n+1}} - \Delta t^2 \frac{\partial \boldsymbol{f}_s}{\partial \boldsymbol{x}}\Big|_{\boldsymbol{x}_k^{n+1}} \tag{3.4}$$

$$\boldsymbol{b}_k^{n+1} = \Delta t \mathcal{M}\big(\boldsymbol{v}^n - \boldsymbol{v}_k^{n+1}\big) + \Delta t^2 \boldsymbol{f}_k^{n+1} \tag{3.5}$$

where the approximation $\frac{\partial \boldsymbol{f}_d}{\partial \boldsymbol{x}} = 0$ was used for efficiency. Positions and velocities are updated after each step as $\boldsymbol{x}_{k+1}^{n+1} = \boldsymbol{x}_k^{n+1} + \Delta\boldsymbol{x}$ and $\boldsymbol{v}_{k+1}^{n+1} = \frac{1}{\Delta t}(\boldsymbol{x}_{k+1}^{n+1} - \boldsymbol{x}^n)$. For quasistatic simulation, the system simplifies to $\mathcal{A}_k^{n+1} = \mathcal{K}(\boldsymbol{x}_k^{n+1})$ and $\boldsymbol{b}_k^{n+1} = \boldsymbol{f}_k^{n+1}$.

In the FEM material discretization, the force Jacobians are the negated damping and stiffness matrices

$$\frac{\partial \boldsymbol{f}_d}{\partial \boldsymbol{v}}\Big|_{\boldsymbol{x}_k^{n+1}} = -\mathcal{C}(\boldsymbol{x}_k^{n+1}) \tag{3.6}$$

$$\frac{\partial \boldsymbol{f}_s}{\partial \boldsymbol{x}}\Big|_{\boldsymbol{x}_k^{n+1}} = -\mathcal{K}(\boldsymbol{x}_k^{n+1}) \tag{3.7}$$

and assuming Rayleigh damping $\mathcal{C} = \alpha\mathcal{M} + \beta\mathcal{K}$, the system matrix becomes

$$\mathcal{A}_k^{n+1} = (1 + \alpha\Delta t)\mathcal{M} + (\beta\Delta t + \Delta t^2)\mathcal{K}(\boldsymbol{x}_k^{n+1}) \tag{3.8}$$

The formulation of Equation (3.3) with unknown displacements $\Delta\boldsymbol{x}$, instead of velocity changes $\Delta\boldsymbol{v}$, as common in the literature [BW98, OTSG09], saves an expensive multiplication by $\mathcal{K}(\boldsymbol{x}_k^{n+1})$ in the right hand term $\boldsymbol{b}_k^{n+1}$ at the cost of an additional multiplication by $\mathcal{M}$, much cheaper due to mass lumping. This saving repeats for every Newton-Raphson iteration k.

Implicit Euler integration is very stable, and in case of linear elastic forces a single Newton-Raphson iteration is generally enough. However, the cost for such stability is numerical energy dissipation, that grows with the timestep [SLM06]. In the context of

interactive simulation with weak accuracy requirements some authors even praise such dissipation [PO09], as it ensures that, in absence of interaction, deformable objects will stop moving rapidly and can be effectively deactivated (*frozen*) to reduce the computational cost.

The exact, fully implicit solution of the nonlinear problem may be computationally expensive. A large number of Newton-Raphson iterations may be necessary, each one requiring the solution of a different system of linear equations assembled from the local evaluation of position-dependent force Jacobians. This is acceptable in the engineering context, where accuracy and validation of simulation results is a priority, but directly contradicts the requirements of interactive simulation detailed in Section 3.1. Fortunately, if we relax the accuracy requirements on the solution, several approximations are possible:

- Use a single Newton-Raphson step in the solution of Equation (3.2)
- Solve the linear system approximately in Equation (3.3).
- Use approximate force Jacobians in Equation (3.4).

The first approximation yields a semi-implicit method, as opposed to the fully-implicit, exact, solution of Equation (3.1) and Equation (3.2). The second and third approximations result in an inexact/modified Newton-Raphson method that is expected to converge, although slower, to the same global solution as the exact version. When both kinds of approximations are used the integration method is quasi-implicit. We will analyze the effects of these approximations in Section 3.3.4.

3.2.1 Solving $\mathcal{A}_k^{n+1}\Delta\boldsymbol{x} = \boldsymbol{b}_k^{n+1}$

The system matrix $\mathcal{A}_k^{n+1}$ is symmetric and very sparse, as $\mathcal{M}$ is diagonal thanks to mass lumping, and $\mathcal{K}(\boldsymbol{x}_k^{n+1})$ only contains non-zero elements for directly adjacent node pairs (i, j) in the finite element discretization. A suitable strategy for the solution of such sparse systems of equations is using iterative Krylov methods [Kel95], that only require matrix-vector products and can be safely stopped when sufficient accuracy is achieved. In realtime applications, it is common to specify a maximum number of iterations n_{LS} to guarantee a strict upper bound on the computation time, at the expense of accuracy. We will analyze the accuracy and computational effects of these approximations in Section 3.3.4.

The approximate solution of $\mathcal{A}_k^{n+1}\Delta\boldsymbol{x} = \boldsymbol{b}_k^{n+1}$ at each iteration k using an iterative linear system solver yields an inexact Newton-Raphson scheme that is expected to maintain good convergence properties [Kel95], and offers a tradeoff between external (Newton-Raphson) and internal (linear system solver) iteration counts.

In addition, for such solvers the system matrix does not need to be explicitly built, only an algorithm to compute the product $\mathcal{A}_k^{n+1}\boldsymbol{y}$ for a given vector $\boldsymbol{y}$ is strictly required. This avoids the need to assemble the full system Jacobian, evaluating instead the directional derivatives along $\delta\boldsymbol{x} = -\boldsymbol{y}$ to obtain $\delta\boldsymbol{f} = \mathcal{K}\boldsymbol{y}$

$$\delta\boldsymbol{f} = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}_k^{n+1}} \delta\boldsymbol{x} = -\mathcal{K}(\boldsymbol{x}_k^{n+1})\delta\boldsymbol{x} = \mathcal{K}\boldsymbol{y} \tag{3.9}$$

and computing the matrix-vector products efficiently as

$$\mathcal{A}_k^{n+1}\boldsymbol{y} = (1 + \alpha\Delta t)\mathcal{M}\boldsymbol{y} - (\beta\Delta t + \Delta t^2)\delta\boldsymbol{f} \tag{3.10}$$

The assembly-free approach minimizes memory requirements and is very efficient for relatively coarse material discretizations, where a sufficiently accurate solution of the system of linear equations can be found in such a small number of iterations that the cost of assembly is not amortized [NPF05]. Additionally, avoiding assembly is convenient for fast prototyping of constitutive material models, as it completely decouples per-element contributions to the global system force and Jacobian, that can be accumulated in any order on the global vectors $\boldsymbol{f}$ and $\delta\boldsymbol{f}$ without requiring closed-form expressions for force Jacobian entries.

If the symmetric matrix $\mathcal{A}_k^{n+1}$ is also positive semi-definite, Conjugate Gradients (CG) is an efficient option in both computational and memory cost. For indefinite systems, MINRES is comparable to CG in terms of computational cost, but requires more memory. For non-symmetric and/or indefinite systems a limited-memory GMRES(m) solver can be used but is considerably more expensive and complex than previous alternatives [SS86]. We use CG for semi-definite systems and MINRES for indefinite ones.

The performance of iterative linear system solvers can be greatly improved by providing an accurate initial guess $\Delta\boldsymbol{x}_0^{n+1}$. In our case we extrapolate a guess for the next displacement from the current velocity $\Delta\boldsymbol{x}_0^{n+1} = \Delta t\boldsymbol{v}^n$.

3.2.2 Force differentials

The force differentials $\delta\boldsymbol{f}_i$ required by implicit integration can be computed by differentiation of Equation (2.53) and Equation (2.54), resulting in

$$\begin{bmatrix}\delta\boldsymbol{f}_2 & \delta\boldsymbol{f}_3 & \delta\boldsymbol{f}_4\end{bmatrix} = -V_e\delta\mathcal{P}\mathcal{D}_m^{-T} \tag{3.11}$$

$$\delta\boldsymbol{f}_1 = -\delta\boldsymbol{f}_2 - \delta\boldsymbol{f}_3 - \delta\boldsymbol{f}_4 \tag{3.12}$$

where *stress differential* $\delta\mathcal{P}(\mathcal{F}, \delta\mathcal{F})$ requires the deformation differential $\delta\mathcal{F} = \delta\mathcal{D}_s\mathcal{D}_m^{-1}$, with $\delta\mathcal{D}_s = \begin{bmatrix}\delta\boldsymbol{x}_{12} & \delta\boldsymbol{x}_{13} & \delta\boldsymbol{x}_{14}\end{bmatrix}$ and $\delta\boldsymbol{x}_{ij} = \delta\boldsymbol{x}_j - \delta\boldsymbol{x}_i$. We also define the per-element vectors $\boldsymbol{x}_e$, $\boldsymbol{r}_e$, $\boldsymbol{f}_e$ from the stacked per-node magnitudes. For implicit integration

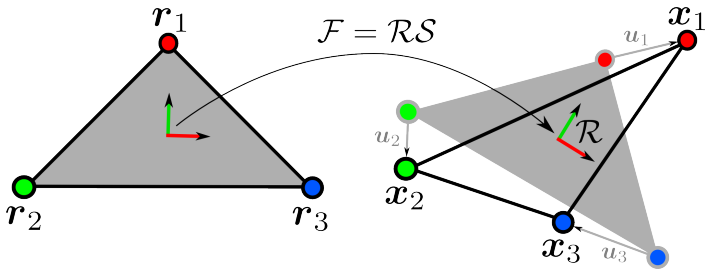

Figure 3.2: Corotational FEM evaluates elastic forces in a rotated coordinate system $\mathcal{R}$ where the deformation $\mathcal{S}$ corresponding to the local node displacements $\boldsymbol{u}_i$ is expected to be small.

without global matrix $\mathcal{A}_k^{n+1}$ assembly, the per-element elastic force differentials $\delta \boldsymbol{f}_e$ can be computed directly from $\delta\mathcal{P}$ and accumulated into the global force differential $\delta \boldsymbol{f}$ without evaluating the element force Jacobian $\frac{\partial \boldsymbol{f}_e}{\partial \boldsymbol{x}_e}\Big|_{\boldsymbol{x}_e}$.

3.3 Corotational linear FEM

The key idea behind corotational linear FEM is the modification of linear elasticity to achieve element-wise rotational invariance, evaluating the deformation in a rotated coordinate frame $\mathcal{R}$ that matches the reference and deformed element configurations [MG04, HS04]. The deformation $\mathcal{F} = \mathcal{RS}$ is decomposed into an orthonormal rotation factor $\mathcal{R}$ and a stretch factor $\mathcal{S}$ that represents local deformation in the rotated frame (Figure 3.2). Even with a linear material model, corotational methods are geometrically non-linear and their fully implicit integration requires the evaluation of exact force differentials as well as a non-linear equation solver. The semi-implicit and quasi-implicit approximations introduced in previous sections are frequently used for interactive simulation and will be thoroughly discussed and compared in the results section.

3.3.1 Element rotations

Given a deformation $\mathcal{F}$, the best matching rotation $\mathcal{R}$ that minimizes $\|\mathcal{F} - \mathcal{R}\|_F^2$ can be computed either by direct *Polar Decomposition* (PD) [SD92, MG04] or as a by-product of the *Singular Value Decompositon* (SVD) [ITF04], and results in a symmetric stretch matrix $\mathcal{S} = \mathcal{R}^T\mathcal{F}$. Alternatively, a fast and robust QR factorization [NPF05, PO09] by Gram-Schmidt orthonormalization can approximate $\mathcal{R}$, but the resulting rotation is not optimal, and as a consequence $\mathcal{S}$ is not symmetric, which induces force anisotropy and limits simulation realism.

For implicit integration, the rotation differential $\delta\mathcal{R}$ is required. For polar decom-

position (and SVD) in 3D it can be compactly expressed as:

$$\delta\mathcal{R} = \mathcal{R}\Big[\mathcal{E} : \Big((\mathrm{tr}(\mathcal{S})\mathcal{I} - \mathcal{S})^{-1}\big(\mathcal{E}^T : (\mathcal{R}^T\delta\mathcal{F})\big)\Big)\Big] \tag{3.13}$$

where $\mathcal{E}$ is the alternating third order tensor that maps vectors to cross product matrices. Further details can be found in [MZS+11a] and its associted Technical Notes, as well as in [CPSS10, Bar12]. The rotation differentials from Equation (3.13) are also correct for the invertible SVD (ISVD) variant introduced in [ITF04]. Although rarely used, the rotation differentials for QR rotation extraction can also be computed analytically from the Gram-Schmidt orthonormalization process.

3.3.2 On the visualization of FEM simplex magnitudes

A simplex finite element in $\mathbb{R}^D$ has $N = D(D+1)$ degrees of freedom, corresponding to the $\mathbb{R}^D$ positions of its $D+1$ vertices, with $N_{2\mathrm{D}} = 6$ and $N_{3\mathrm{D}} = 12$. Displaying information in such highly-dimensional configuration spaces in a meaningful, understandable way is highly challenging. In our discussion, intuition on the qualitative behaviour of a given constitutive material model will be gained from visualization of the *qualitative shape* of energy and force fields for 2D simplex elements displayed as scalar and vector fields in a reduced configuration domain. From the full 6 DOF of a 2D simplex, 2 DOF can be discarded due to translation invariance, and 1 DOF due to rotation invariance. While elastic energy and force are not invariant to uniform scaling, an additional DOF can be discarded if we ignore it. Uniform scaling plays a minor role in most applications of FEM, as objects mainly rotate, bend, stretch and squash but rarely expand or contract uniformly beyond a small fraction. Thus, we can reduce the configuration space to $\mathbb{R}^2$ if we assume two vertex positions constant $\boldsymbol{x}_2 = \boldsymbol{r}_2$ and $\boldsymbol{x}_3 = \boldsymbol{r}_3$, and use the position of the remaining free vertex $\boldsymbol{x}_1$ as the DOF. In practice, we will limit the visualization to a square region centered at the origin $\boldsymbol{x}_1 \in [-h, h]^2$, and use the *reference triangle element* shown in Figure 3.3 in several plots throught this chapter. This will allow us to display the elastic energy Ψ, the elastic forces $\boldsymbol{f}_1$ and other related magnitudes as fields on a bidimensional domain. See, for example, Figure 3.5 and Figure 4.2.

Energy and force field visualization in the principal stretch domain was used in [SHST12] to analyze and communicate the qualitative dynamics of a simplex element in 2D. For our purposes, however, such a representation does not provide sufficient intuition on the actual node trajectories, that will be essential in the discussion of invertible FEM in Section 4.

3.3.3 Stress and force differentials

Recalling the energy and stress expressions for the Linear Corotated Material (LCM) from Section 2.1.4.2, the *exact* force differentials $\delta\boldsymbol{f}$ required by implicit integration

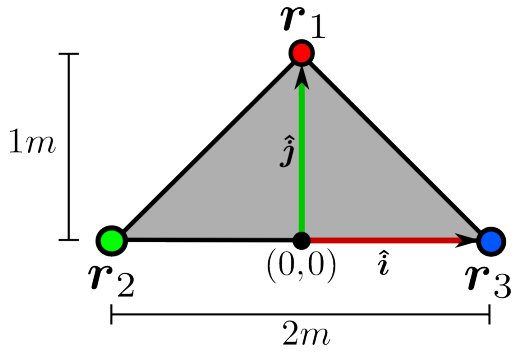

Figure 3.3: Reference triangle element geometry, used in all energy and force field visualizations throughout Chapter 3 and Chapter 4.

can be computed from the stress differentials Equation (3.16)

$$\Psi_C = \mu\|\epsilon_C\|_F^2 + \frac{\lambda}{2}\operatorname{tr}^2(\epsilon_C) \tag{3.14}$$

$$\mathcal{P}_C = \mathcal{R}[2\mu\epsilon_C + \lambda\operatorname{tr}(\epsilon_C)\mathcal{I}] \tag{3.15}$$

$$\delta\mathcal{P}_C = 2\mu\delta\mathcal{F} + \lambda\operatorname{tr}(\mathcal{R}^T\delta\mathcal{F})\mathcal{R} + [\lambda\operatorname{tr}(\epsilon_C) - 2\mu]\delta\mathcal{R} \tag{3.16}$$

Equivalently, corotational linear FEM forces and their exact differentials can be computed using the scheme introduced in [MG04], which evaluates linear elastic forces in the local element frame and rotates them to the global frame afterwards

$$\boldsymbol{f}_e = -\mathcal{R}_e\mathcal{K}_e^0\boldsymbol{u}_e \tag{3.17}$$

$$\delta\boldsymbol{f}_e = -\mathcal{R}_e\mathcal{K}_e^0(\mathcal{R}_e^T\delta\boldsymbol{x}_e + \delta\mathcal{R}_e^T\boldsymbol{x}_e) - \delta\mathcal{R}_e\mathcal{K}_e^0\boldsymbol{u}_e \tag{3.18}$$

where $\mathcal{K}_e^0$ is the 12×12 constant stiffness matrix from linear elasticity, $\mathcal{R}_e$ and $\delta\mathcal{R}_e$ contain $\mathcal{R}$ and $\delta\mathcal{R}$ blocks along the diagonal respectively, and $\boldsymbol{u}_e = \mathcal{R}_e^T\boldsymbol{x}_e - \boldsymbol{r}_e$.

3.3.4 Approximate stress differentials

The exact force differentials obtained through $\delta\mathcal{P}_C$ from Equation (3.16) or directly from Equation (3.18) yield a symmetric but possibly indefinite global stiffness matrix $\mathcal{K}$. Several approximations have been proposed to ensure the positive semi-definiteness of $\mathcal{K}$, which allows the use of fast iterative methods such as Conjugate Gradients. Approximation of force differentials does not change the solution of the problem, only the intermediate Newton-Raphson iterates $\Delta\boldsymbol{x}_k^{n+1}$, that will converge slower to the same global solution. Previously proposed approximations can be classified in two categories: *truncation* and *projection*.

The simplest approximations truncate the force differentials by assuming a constant $\mathcal{R}$ during each Newton-Raphson step, with $\delta\mathcal{R} = 0$. This was the de-facto standard

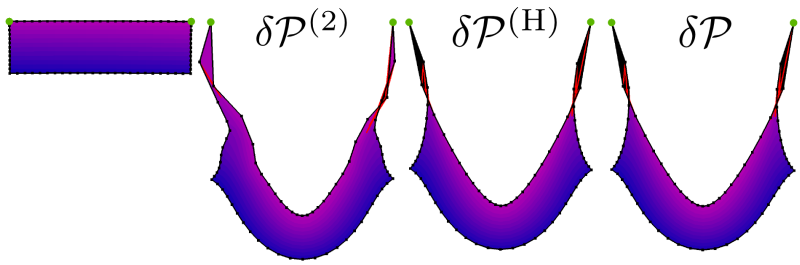

Figure 3.4: A quasi-statically simulated beam fixed at the green nodes and attracted gravity reaches a static equilibrium symmetric configuration when using exact and hybrid stress differentials, but yields alternating, unstable configurations for $\delta\mathcal{P}^{(2)}$ that does never converge.

approach for corotational linear FEM [MG04, NPF05, PO09], until the effects of inaccurate force Jacobians were widely reported [CPSS10, MZS+11a, Bar12]. In order to overcome the inaccuracy of truncated differentials while ensuring positive-definiteness of $\mathcal{K}$, two projection methods have been proposed in [TSIF05, MZS+11a]. Both methods correct per-element force Jacobians by clamping their negative eigenvalues to 0, which conservatively ensures the positive-definiteness of the assembled $\mathcal{K}$ for any constitutive material model. This was achieved by means of a per-element diagonalization [TSIF05] or Singular Value Decomposition [MZS+11a] of a 3×3 matrix. The resulting force differentials are more accurate, but also more expensive to compute, than truncated ones.

Despite its lower accurary, truncation is best suited to meet the strict efficiency requirements of interactive simulation, as it not only completely avoids computing the $\delta\mathcal{R}$, but also cancels many terms in Equation (3.16) and Equation (3.18). We will analyze the inaccuracy induced by truncation and propose an efficient method to improve its results in the context of interactive simulation. Assuming $\delta\mathcal{R} = 0$ in Equation (3.16) yields

$$\delta\mathcal{P}^{(1)} = 2\mu\delta\mathcal{F} + \lambda\,\mathrm{tr}(\mathcal{R}^T\delta\mathcal{F})\mathcal{R} \tag{3.19}$$

The same assumption for Equation (3.18) results in

$$\delta\boldsymbol{f}_e = -\mathcal{R}_e\mathcal{K}_e^0\mathcal{R}_e^T\delta\boldsymbol{x}_e \tag{3.20}$$

which, according to [MZS+11a], is equivalent to the stress differential

$$\delta\mathcal{P}^{(2)} = 2\mu\mathcal{R}(\mathcal{R}^T\delta\mathcal{F})_{\mathrm{sym}} + \lambda\,\mathrm{tr}(\mathcal{R}^T\delta\mathcal{F})\mathcal{R} \tag{3.21}$$

with $\mathcal{A}_{\mathrm{sym}} = \frac{1}{2}(\mathcal{A} + \mathcal{A}^T)$. The approximation in Equation (3.20) yields the element stiffness matrix $\mathcal{K}_e^R = \mathcal{R}_e\mathcal{K}_e^0\mathcal{R}_e^T$ often used in corotational linear FEM [MG04]. Despite Equation (3.16) and Equation (3.18) being equivalent, the approximations Equation (3.19) and Equation (3.21) are not. Both are considerably cheaper to compute

than their exact counterparts, and guarantee a symmetric and positive semi-definite global $\mathcal{K}$. However, as shown in Figure 3.5, $\delta\mathcal{P}^{(2)}$ is very inaccurate for large deformations, while $\delta\mathcal{P}^{(1)}$ is less accurate for small deformations. Stress differential inaccuracy in case of large deformations leads to instabilities in the Newton-Raphson solution that cause node vibration between several distant solutions [MZS+11a], as shown in Figure 3.4. This is specially serious in quasistatic simulations, where there are no inertia terms in $\mathcal{A}_k^{n+1}$ that can reduce the impact of inaccuracies in $\mathcal{K}$. Stress differential inaccuracy in case of small deformations leads to increased numerical damping in the Backwards-Euler solution, which results in insufficient momentum conservation for moderate deformations, as shown in the corresponding column of Figure 3.6.

We propose a *hybrid stress approximation* that retains the accuracy of $\delta\mathcal{P}^{(2)}$ for small deformations but gradually tends to $\delta\mathcal{P}^{(1)}$ for large ones. Comparing Equation (3.19) with Equation (3.21), the only difference is in the first summand, which we interpolate as

$$\gamma = \max\left(0, \min\left(\frac{\mathrm{tr}(|\mathcal{S} - \mathcal{I}|)}{D}, 1\right)\right) \tag{3.22}$$

$$\delta\mathcal{P}^{(\mathrm{H})} = \gamma\,\delta\mathcal{P}^{(1)} + (1-\gamma)\,\delta\mathcal{P}^{(2)} \tag{3.23}$$

where D is the number of spatial dimensions considered. The interpolation weight γ measures the average stretch ratio deviation from its reference value of 1. The convex combination of $\delta\mathcal{P}^{(1)}$ and $\delta\mathcal{P}^{(2)}$ preserves the symmetry and positive definiteness of their combined force Jacobians. This approximation is computationally efficient and yields plausible results for a wide range of deformation magnitudes (Figure 3.5). However, as the Poisson ratio ν increases, the λ-terms common to $\delta\mathcal{P}^{(1)}$ and $\delta\mathcal{P}^{(2)}$ dominate over the μ-terms, and both approximations converge to the same highly inaccurate result, as seen in the last row of Figure 3.5. Nevertheless, $\delta\mathcal{P}^{(\mathrm{H})}$ remains advantageous for $\nu < 0.4$. Large ν are known to be problematic for linear constitutive models, as they represent material incompressibility that is more accurately modelled by nonlinear constitutive models (eg. neo-Hookean) that enforce $\det(J) = 1$ [SHST12].

3.3.5 Results

We performed a set of experiements with simple beam models described in Table 3.1 to analyze the behaviour of each stress differential definition and its effect on the dynamics and computational cost. In order to analyze energy conservation no damping was used ($\alpha = \beta = 0$). The beams are fixed by their upper-left corner and subject to a strong gravity acceleration $g = -100m/s^2$.

For each experiment we ran the simulation 4×4 times during $5s$ for each of the 4 stress differential definitions $\{\delta\mathcal{P}, \delta\mathcal{P}^{(1)}, \delta\mathcal{P}^{(2)}, \delta\mathcal{P}^{(\mathrm{H})}\}$, exploring the numerical inte-

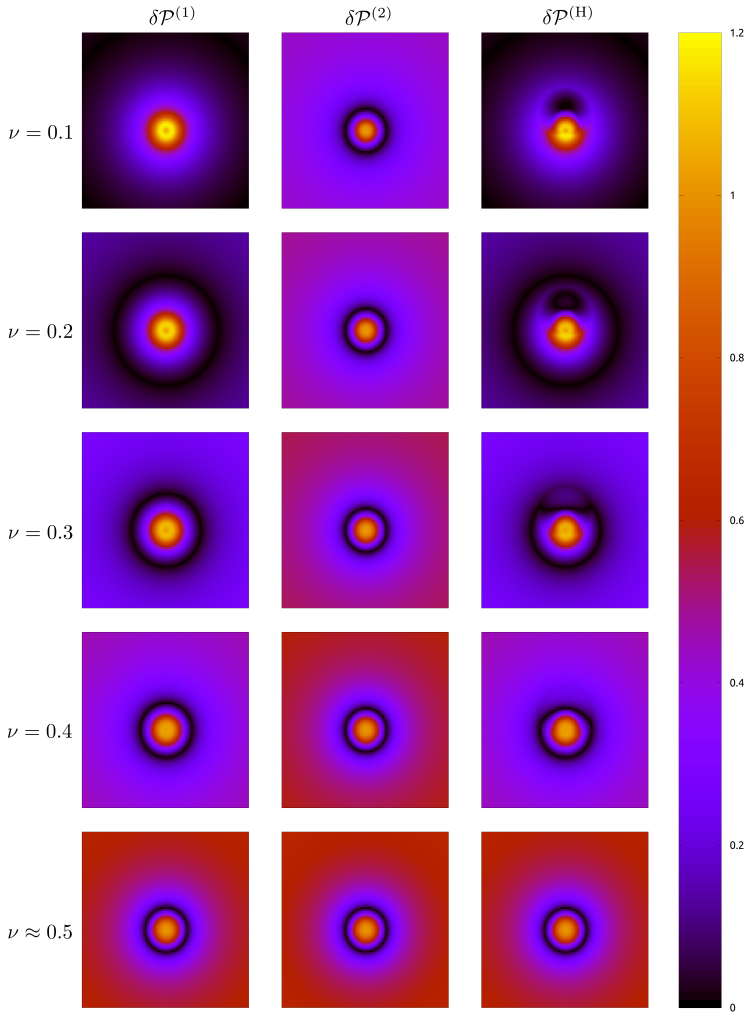

Figure 3.5: We plot the relative error in the approximate force Jacobians induced by displacement of node $\boldsymbol{x}_1$ of the reference triangle element in a $[-8, 8]^2$ region of the plane, with $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$ kept in place. Left column shows the error for $\delta\mathcal{P}^{(1)}$ Equation (3.19), middle column for $\delta\mathcal{P}^{(2)}$ Equation (3.20) and right column for $\delta\mathcal{P}^{(\mathrm{H})}$ Equation (3.23). Rows show increasing values of the Poisson ratio ν for a Linear Corotated Material with $E = 1000$ and invertible SVD rotation extraction [ITF04]. Exact $\delta\mathcal{P}$ Equation (3.16) used as reference to compute relative error as $\|\delta\mathcal{P}^{(\mathrm{approx})} - \delta\mathcal{P}\|_F / \|\delta\mathcal{P}\|_F$.

gration parameter space for the timestep $\Delta t = \{\frac{1}{30}, \frac{1}{60}, \frac{1}{120}, \frac{1}{240}\}$ and the maximum number of Newton-Raphson iterations $n_{\mathrm{NR}} = \{1, 2, 4, 8\}$.

All experiments used ISVD rotation extraction, and a Conjugate Gradients linear system solver with a target relative precision of 0.001, except the ones involving $\delta\mathcal{P}$, that use MINRES in order to handle a potentially indefinite system matrix $\mathcal{A}_k^{n+1}$.

	Dimensions	mass	E	ν	Elements	Nodes
Beam2D	$2.1m \times 0.6m$	$5kg$	$1000Pa$	0.3	576	325
Beam3D	$2.1m \times 0.6m \times 0.6m$	$1.6kg$	$1000Pa$	0.35	324	112

Table 3.1: Models used in stress differential approximation benchmarks.

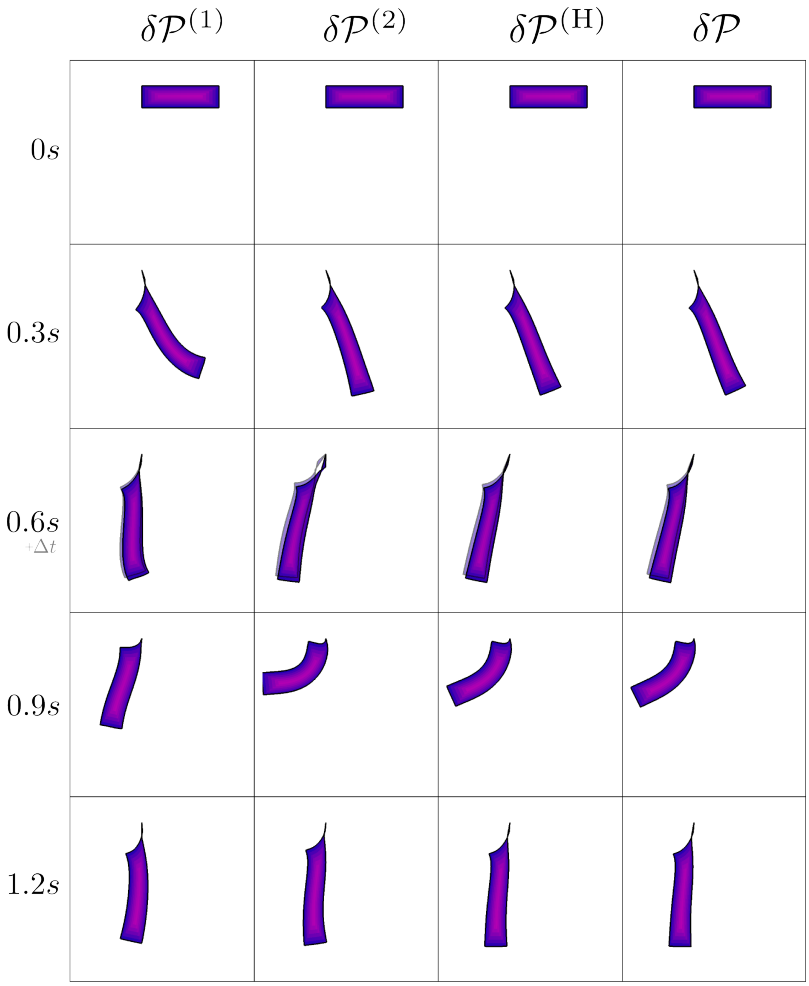

Figure 3.6: Simulation snapshots for the test beam in Experiment1 with $\Delta t = \frac{1}{60}s$ and $n_{\mathrm{LS}} = \infty$. Middle row shows two consecutive configurations at $t = 60s$ and $t = 60s + \Delta t$ overlayed, where the instability of $\delta\mathcal{P}^{(2)}$ can be appreciated in the most elongated part of the model.

Experiment1: In the first experiment, we simulated Beam2D with an unlimited number of linear solver iterations $n_{\mathrm{LS}} = \infty$. Simulation snapshots for specific parameter values can be seen in Figure 3.6. The individual results of Experiment1 for each stress differential definition for all parameter combinations can be seen in Figure 3.7, Figure 3.8, Figure 3.9, Figure 3.10, and summarized in Figure 3.11 and Figure 3.12.

The results for exact $\delta\mathcal{P}$ (Figure 3.7) show that energy conservation greatly improves as Δt decreases (vertical), but does not benefit from additional NR iterations (horizontal), which signals that the actual nonlinear problem Equation (3.2) is being

accurately solved in a single NR iteration thanks to the exact force Jacobian, but that the intrinsic energy disspation of Implicit Euler integration can only be avoided by reducing the timestep.

Energy conservation with approximate differentials $\delta\mathcal{P}^{(1)}$ (Figure 3.8) does benefit from additional NR iterations (horizontal), and only matches the exact results a a given value of Δt if at least 4 NR iterations are performed. This observation matches the expected behaviour of modified Newton-Raphson solvers. Again, reducing Δt (vertical) improves energy conservation.

A completely different situation arises with $\delta\mathcal{P}^{(2)}$ (Figure 3.9). Overestimated stress differentials for large stretches result in periodic energy fluctuations caused by alternating unstable solutions of the problem Equation (3.2), a fact amplified by Newton-Raphson iterations. Only decreasing the timestep minimizes these instabilities, as it reduces the effect of overestimation errors.

Finally, the results for our proposed $\delta\mathcal{P}^{(\mathrm{H})}$ shown in (Figure 3.9) closely match the results for $\delta\mathcal{P}$ across the whole 4×4 parameter space, signaling a very accurate approximation to the exact stress differentials. The summarized 4-way comparison of energy conservation and accumulated CPU cost for all stress differentials can be seen in Figure 3.11 and Figure 3.12 respectively, and show a clear global trend: $\delta\mathcal{P}^{(\mathrm{H})}$ matches the energy conservation results of $\delta\mathcal{P}$ at approximately 50% of the CPU cost. The higher cost of $\delta\mathcal{P}$ is mostly caused by the evaluation of extra terms including $\delta\mathcal{R}$ ($\approx 90\%$) but also by the use of a MINRES linear solver instead of CG ($\approx 10\%$). When compared with $\delta\mathcal{P}^{(1)}$ and $\delta\mathcal{P}^{(2)}$, the cost of $\delta\mathcal{P}^{(\mathrm{H})}$ is slightly higher, but yields much better stability and energy conservation.

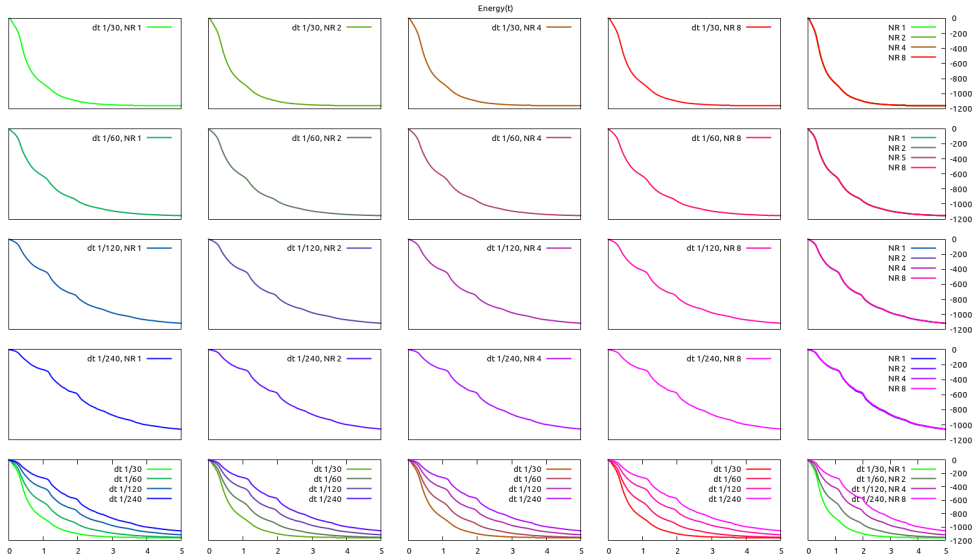

Figure 3.7: *(Experiment1)* Mechanical energy for exact $\delta\mathcal{P}$. The 4×4 upper-left plots explore the parameter space. Plots in the lower 4×1 row, right 1×4 column and lower-right corner show summarized results for their column, row and the diagonal, respectively.

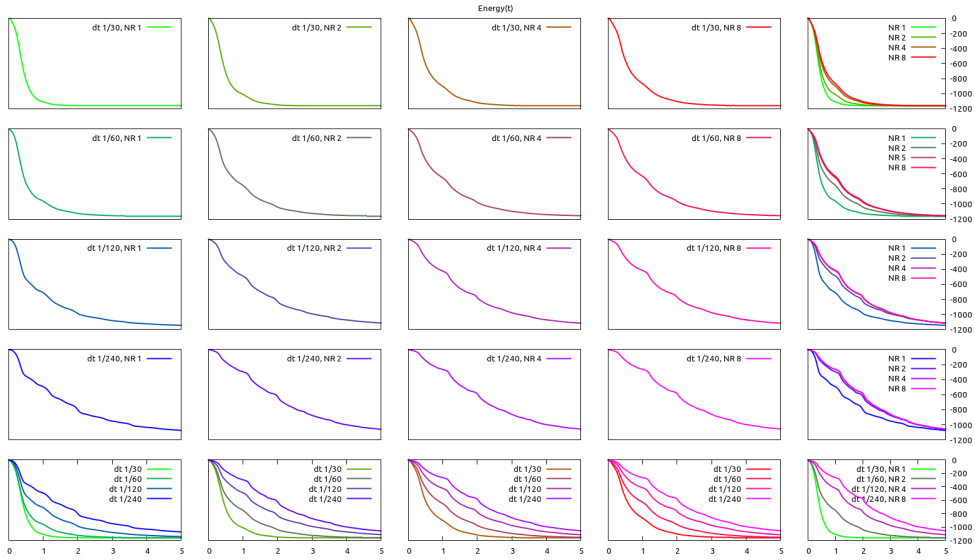

Figure 3.8: *(Experiment1)* Mechanical energy for $\delta\mathcal{P}^{(1)}$. The 4×4 upper-left plots explore the parameter space. Plots in the lower 4×1 row, right 1×4 column and lower-right corner show summarized results for their column, row and the diagonal, respectively.

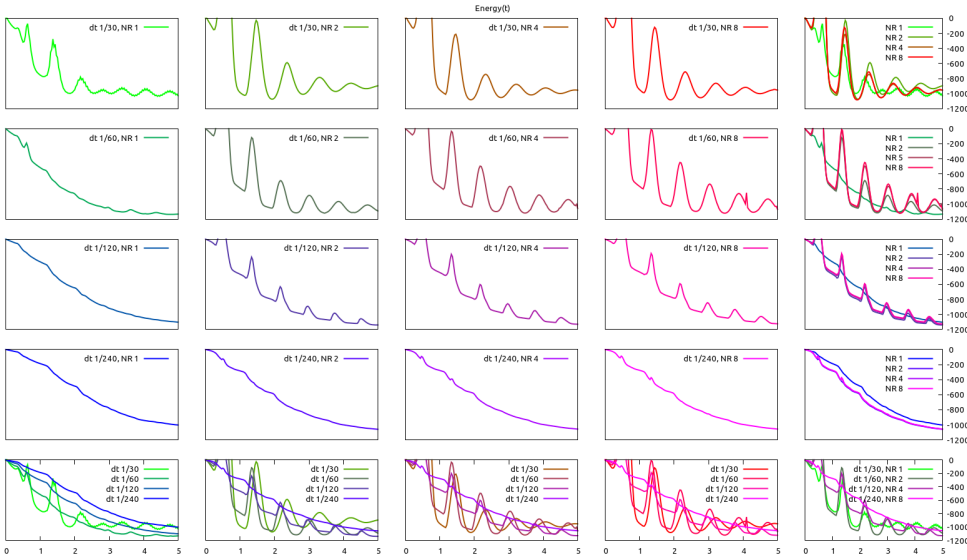

Figure 3.9: *(Experiment1)* Mechanical energy for $\delta\mathcal{P}^{(2)}$. The 4×4 upper-left plots explore the parameter space. Plots in the lower 4×1 row, right 1×4 column and lower-right corner show summarized results for their column, row and the diagonal, respectively.

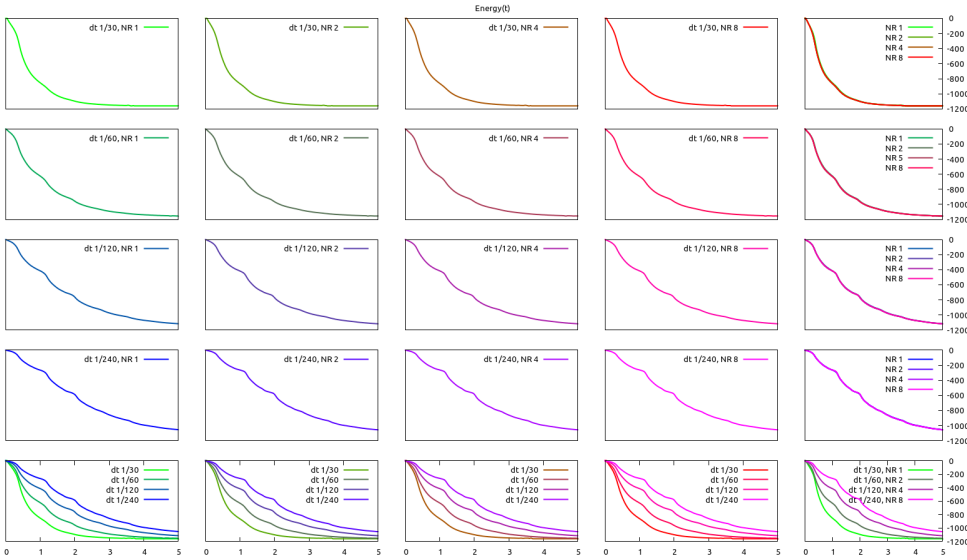

Figure 3.10: *(Experiment1)* Mechanical energy for $\delta\mathcal{P}^{(\mathrm{H})}$. The 4×4 upper-left plots explore the parameter space. Plots in the lower 4×1 row, right 1×4 column and lower-right corner show summarized results for their column, row and the diagonal, respectively.

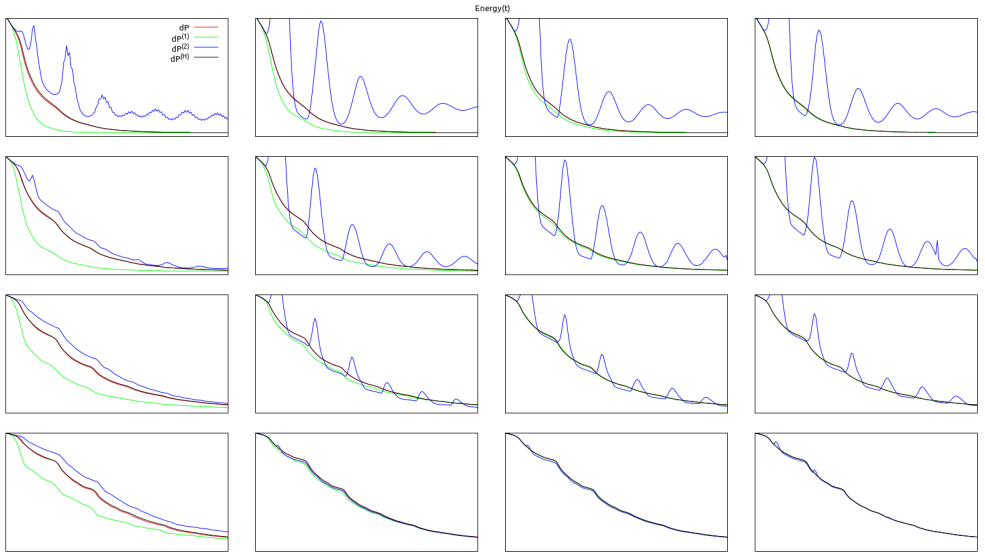

Figure 3.11: *(Experiment1)* 4-way energy conservation comparison between $\delta\mathcal{P}$, $\delta\mathcal{P}^{(1)}$, $\delta\mathcal{P}^{(2)}$, $\delta\mathcal{P}^{(\mathrm{H})}$ for the same $\Delta t \times n_{\mathrm{NR}}$ parameter space in Figures 3.7..3.10. Unlimited linear system solver iterations. All plots use the same scales.

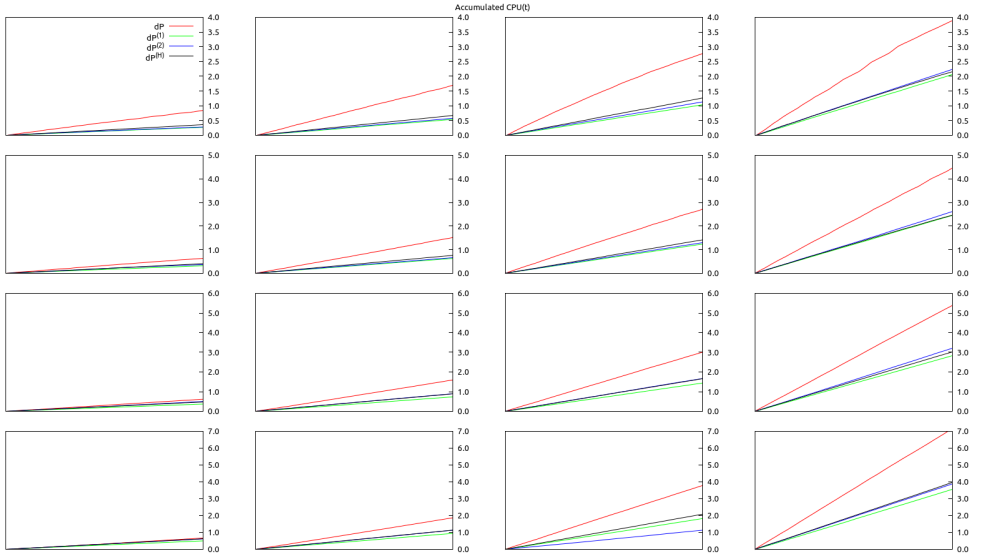

Figure 3.12: *(Experiment1)* 4-way accumulated CPU cost comparison between $\delta\mathcal{P}$, $\delta\mathcal{P}^{(1)}$, $\delta\mathcal{P}^{(2)}$, $\delta\mathcal{P}^{(\mathrm{H})}$ for the same $\Delta t \times n_{\mathrm{NR}}$ parameter space in Figures 3.7..3.10. Unlimited linear system solver iterations. All plots use the same X scale and different Y scale for each Δt value (rows).

Experiment2: The second experiment simulates Beam2D with limited linear system solver iterations $n_{\text{LS}} = 10$. This is a common practice in production-oriented interactive simulations (eg. videogames), where the total computation time per frame is required to be under strict control, and fluctuations are highly undesirable as they may affect the frame-rate perceived by the user. The results are directly summarized in Figure 3.13 and Figure 3.14. The relative effect of each stress approximation is very similar to Experiment1 in terms of energy conservation and CPU cost. Using the energy conservation of $\delta\mathcal{P}$ in Experiment1 as a reference, we see that when $n_{\text{LS}} = 10$, for larger timesteps $\Delta t = \{\frac{1}{30}, \frac{1}{60}\}$ (upper rows in Figure 3.13) it takes up to 8 Newton-Raphson iterations to match the reference results, even using the exact $\delta\mathcal{P}$. For smaller $\Delta t = \{\frac{1}{120}, \frac{1}{240}\}$, however, the iteration limit has a very small impact, and all stress definitions produce analogous results to Experiment1.

Experiment3: We simulate Beam3D with $n_{\text{LS}} = 10$. The results are directly summarized in Figure 3.15 and Figure 3.16. Although instabilities are less dramatic, results are qualitatively similar to the two-dimensional Experiment2, and analogous conclusions can be drawn.

Experiment4: We simulate Beam3D with increasing values of Young modulus $E = \{10^3, 10^4, 10^5, 10^6\}Pa$ and a different parameter space for maximum linear solver iterations $n_{\text{LS}} = \{10, 20, 40, \infty\}$ (rows) and Newton-Raphson iterations $n_{\text{NR}} = \{1, 2, 4, 8\}$ (columns), using $\delta\mathcal{P}^{(\text{H})}$ with $\nu = 0.35$ in all cases. The results for all values of E are summarized in the potential energy plot in Figure 3.17 for $\Delta t = \frac{1}{60}s$ and Figure 3.18 for $\Delta t = \frac{1}{120}s$. As stiffness increases, the results of insufficient convergence due to limited linear system solver or Newton-Raphson iterations manifest as a decrease in the oscillation frequency and the observed gravity acceleration, as detailed in [SLM06]. Insufficient convergence results in a visually unnatural slow-motion movement. As the number of iterations grows, the oscillation frequency converges to the correct one, while still affected by intrinsic numerical damping due to Implicit Euler integration. Figure 3.17 suggests that for stiff linear materials increasing n_{LS} (rows) is more efficient than increasing n_{NR} (columns). Comparing Figure 3.17 and Figure 3.18 shows that reducing the timestep also improves stiff material behaviour and energy conservation, as in previous experiments.

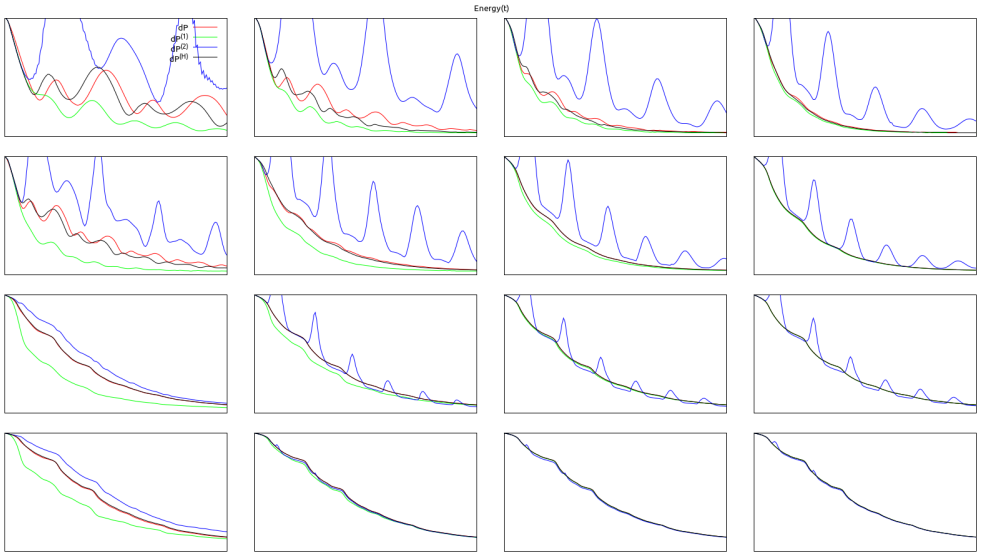

Figure 3.13: *(Experiment2)* 4-way energy conservation comparison between $\delta\mathcal{P}$, $\delta\mathcal{P}^{(1)}$, $\delta\mathcal{P}^{(2)}$, $\delta\mathcal{P}^{(\mathrm{H})}$ for the same $\Delta t \times n_{\mathrm{NR}}$ parameter space in Figures 3.7..3.10. Limited linear solver iterations $n_{\mathrm{LS}} = 10$. All plots use the same scales.

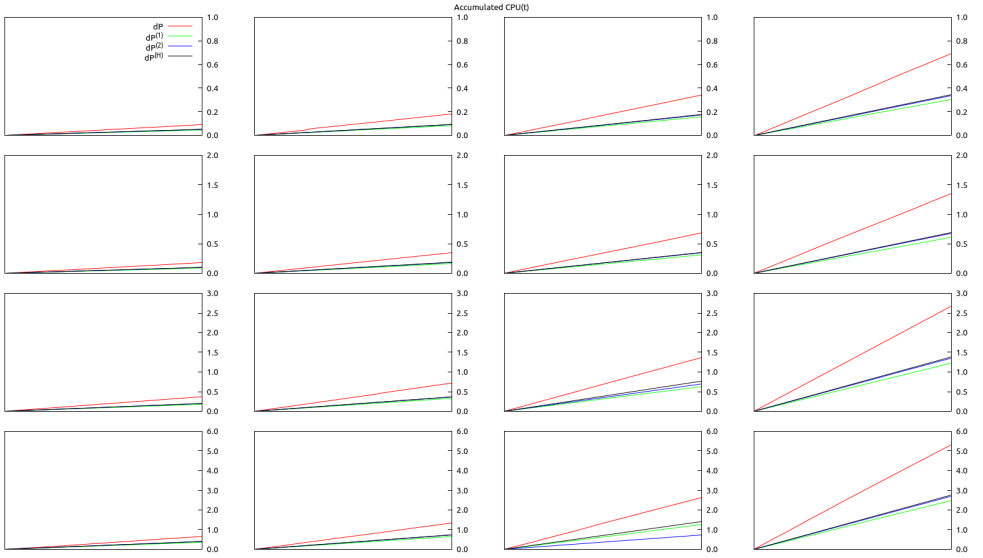

Figure 3.14: *(Experiment2)* 4-way accumulated CPU cost comparison between $\delta\mathcal{P}$, $\delta\mathcal{P}^{(1)}$, $\delta\mathcal{P}^{(2)}$, $\delta\mathcal{P}^{(\mathrm{H})}$ for the same $\Delta t \times n_{\mathrm{NR}}$ parameter space in Figures 3.7..3.10. Limited linear solver iterations $n_{\mathrm{LS}} = 10$. All plots use the same X scale and different Y scale for each Δt value (rows).

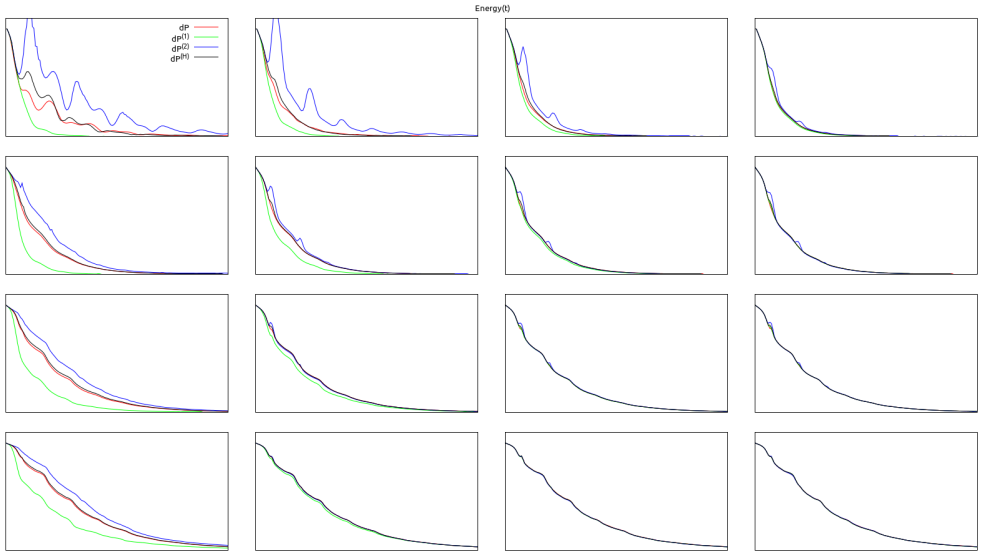

Figure 3.15: *(Experiment3)* 4-way energy conservation comparison between $\delta\mathcal{P}$, $\delta\mathcal{P}^{(1)}$, $\delta\mathcal{P}^{(2)}$, $\delta\mathcal{P}^{(\mathrm{H})}$ for the same $\Delta t \times n_{\mathrm{NR}}$ parameter space in Figures 3.7..3.10. Limited linear solver iterations $n_{\mathrm{LS}} = 10$. All plots use the same scales.

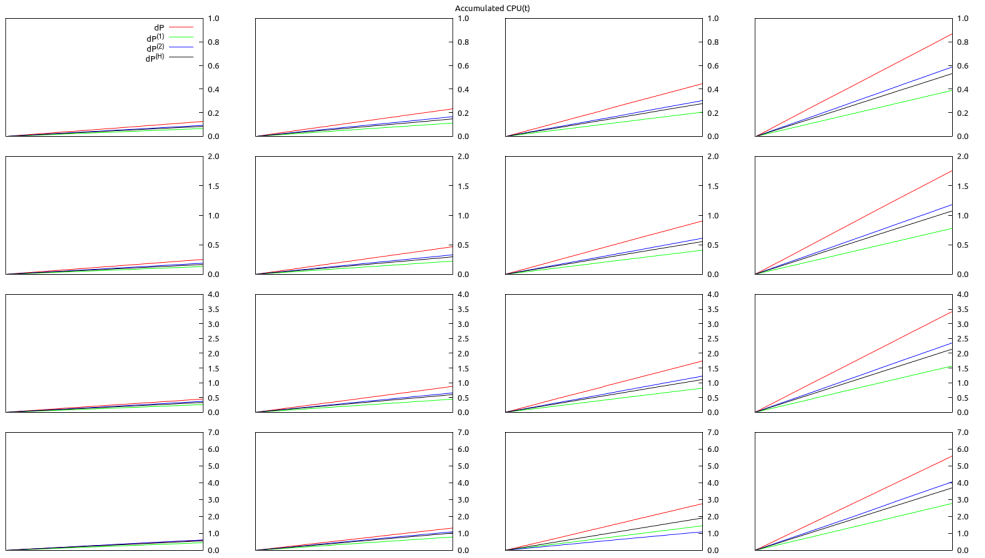

Figure 3.16: *(Experiment3)* 4-way CPU cost comparison between $\delta\mathcal{P}$, $\delta\mathcal{P}^{(1)}$, $\delta\mathcal{P}^{(2)}$, $\delta\mathcal{P}^{(\mathrm{H})}$ for the same $\Delta t \times n_{\mathrm{NR}}$ parameter space in Figures 3.7..3.10. Limited linear solver iterations $n_{\mathrm{LS}} = 10$. All plots use the same X scale and different Y scale for each Δt value (rows).

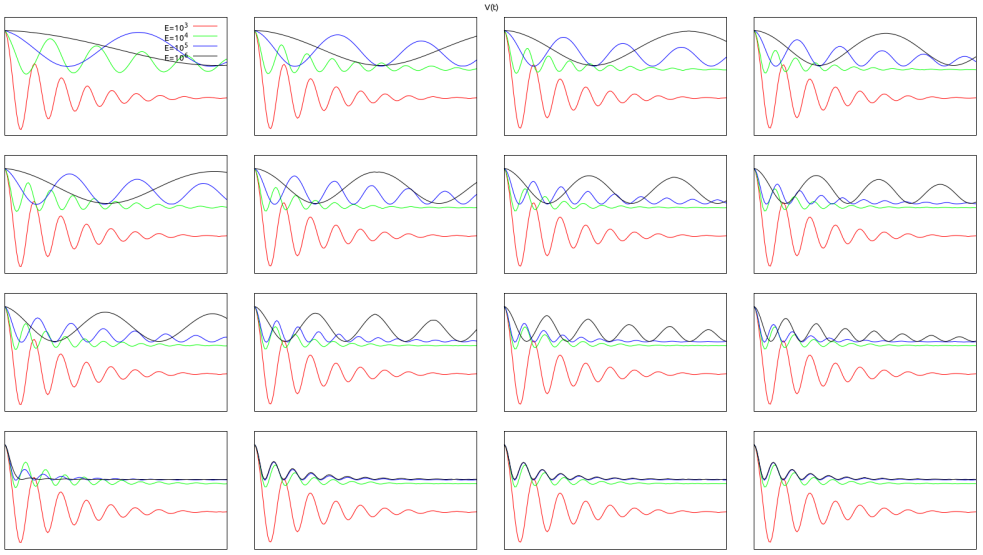

Figure 3.17: *(Experiment4)* 4-way gravitational potential energy comparison for different Young modulus values $E = \{10^3, 10^4, 10^5, 10^6\}Pa$ in a $n_{\text{LS}} \times n_{\text{NR}}$ (rows x columns) parameter space $\{10, 20, 40, \infty\} \times \{1, 2, 4, 8\}$, and $\Delta t = \frac{1}{60}s$. All examples use $\delta\mathcal{P}^{(\text{H})}$ and the same plot scales.

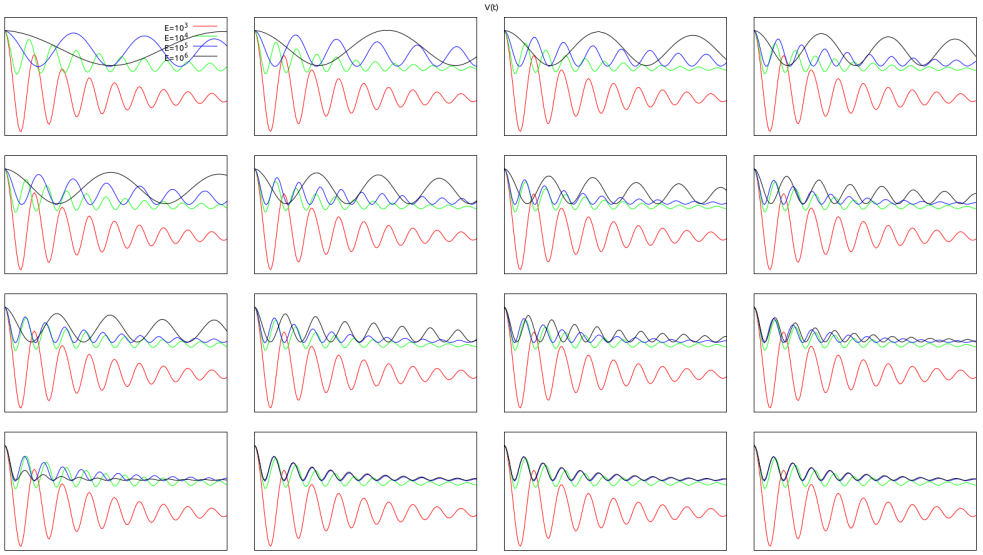

Figure 3.18: *(Experiment4)* 4-way gravitational potential energy comparison for different Young modulus values $E = \{10^3, 10^4, 10^5, 10^6\}Pa$ in a $n_{\text{LS}} \times n_{\text{NR}}$ (rows x columns) parameter space $\{10, 20, 40, \infty\} \times \{1, 2, 4, 8\}$, and $\Delta t = \frac{1}{120}s$. All examples use $\delta\mathcal{P}^{(\text{H})}$ and the same plot scales.

3.3.6 Discussion

The proposed hybrid stress differential approximation $\delta\mathcal{P}^{(\mathrm{H})}$ is not based in physical principles but in mathematical reasoning and observation of the *effects* of the approximation errors. In particular, the interpolation paramater γ does not necessarily represent any conventional continuum mechanics magnitude. However, we believe that the presented results fully justify its use in interactive simulation to improve the accuracy of approximate force Jacobians with very small computational overhead.

Given a fixed amount of computational resources to solve the dynamics equations Equation (3.1), Equation (3.2), a fundamental question emerges: which are the optimal values of Δt, n_{NR} and n_{LS}? Experiments 1,2 and 3 suggest that if we use $\delta\mathcal{P}$ or $\delta\mathcal{P}^{(\mathrm{H})}$, reducing the timestep is better than increasing the number of NR iterations, both in terms of solution stability and energy conservation. The optimal parameters among the tested combinations are $\Delta t = \frac{1}{120}$ and $n_{\mathrm{NR}} = 1$. We do not claim this conclusion to be universal, however, as many other factors can affect the overall efficiency of a simulation scheme, such as the required accuracy, the individual characteristics of the simulated models, the optimization/efficiency of the implementation, or even specific hardware details (arithmetic and memory speed, etc). We propose using the previous experimental strategy to explore the numerical integration solver parameter space automatically. In a production environment, such as the development of a videogame or an interactive simulation application, this methodology could be used to determine suitable solver parameter values automatically in a preprocess.

Realistic simulation of stiff materials with Implicit Euler significantly increases the CPU cost, as it requires a large number of iterations, or a smaller timestep, to converge properly. Alternative formulations such as constraint-based elasticity should be explored in this case [SLM06, TNGF15].

3.4 Interaction

So far we have described the dynamics of a free deformable body subject to external gravity with optionally static nodes, as in Figure 3.6. In interactive simulation the free body equations of motion can be *transiently* changed. The nature of such changes (interactions) is potentially diverse, and includes the application of external forces and tractions, direct control over node positions, and automatic contact treatment, to name a few. For some applications, even unphysical interactions may be required (eg. a videogame may include in its gameplay mechanics the ability to modify the mass of a game object, change its material properties, or temporally disable gravity). A general purpose interactive simulation scheme must allow as much interaction freedom as possible and, ideally, remain efficient and stable under all circumstances.

We emphasize the transient, unpredictable nature of interaction, and its on-line influence on the simulation process, that generates a feedback loop. In the next sections we will describe several general purpose *interaction primitives* on deformable solids discretized into simplex elements. A specific application may compose and apply them in any way according to its needs. The interaction primitives considered will be

Forces and impulses: Forces are the most natural interactions, as they fully preserve the structure of the dynamics equations (eg. gravity and user-defined springs).

Position and velocity changes: While unphysical, direct manipulation of positions and velocities is useful for imperative control of dynamics objects.

Geometric constraints: The definition of geometric constraints allows physically consistent indirect control on the simulated objects, such as attaching a material point to static geometry.

Contact constraints: A suitable strategy to avoid solid geometry interpenetration in interactive simulations is using inequality contact constraints that are automatically created in the contact determination phase (detailed in Chapter 5).

3.4.1 Forces, impulses, position and velocity changes

Within the framework presented in [WGW90] we will treat an embedded point $\boldsymbol{p}$ as a *connector* on the tetrahedral element nodes $\boldsymbol{p}(\boldsymbol{x}_e)$, and show how to apply forces, impulses, position and velocity changes on it. Using the constant barycentric coordinates

ξ_i we obtain the following kinematic relationships

$$\boldsymbol{p} = \sum_{i=1}^{4} \xi_i \boldsymbol{x}_i = \mathcal{J}\boldsymbol{x}_e \quad , \quad \dot{\boldsymbol{p}} = \mathcal{J}\dot{\boldsymbol{x}}_e \quad , \quad \ddot{\boldsymbol{p}} = \mathcal{J}\ddot{\boldsymbol{x}}_e \tag{3.24}$$

$$\mathcal{J} = \frac{\partial \boldsymbol{p}}{\partial \boldsymbol{x}_e} = \begin{bmatrix} \xi_1 & 0 & 0 & \xi_2 & 0 & 0 & \xi_3 & 0 & 0 & \xi_4 & 0 & 0 \\ 0 & \xi_1 & 0 & 0 & \xi_2 & 0 & 0 & \xi_3 & 0 & 0 & \xi_4 & 0 \\ 0 & 0 & \xi_1 & 0 & 0 & \xi_2 & 0 & 0 & \xi_3 & 0 & 0 & \xi_4 \end{bmatrix} \tag{3.25}$$

From virtual work equivalence [NKJF09], forces $\boldsymbol{f}_p$ and impulses $\boldsymbol{j}_p$ applied at an arbitrary point $\boldsymbol{p}$ in a simplex element are distributed on nodes as

$$\boldsymbol{f}_e = \mathcal{J}^T \boldsymbol{f}_p \tag{3.26}$$

$$\boldsymbol{j}_e = \mathcal{J}^T \boldsymbol{j}_p \tag{3.27}$$

where impulses can be considered time-integrated forces $\boldsymbol{j} = \Delta t \boldsymbol{f}$.

We also need the formulas to distribute a change of position $\Delta\boldsymbol{p}$ and velocity $\Delta\dot{\boldsymbol{p}}$ at the embedded point on the element nodes $\boldsymbol{x}_i$. The results will match those presented in [SSIF07, NKJF09], but we derive them here for completeness. Assuming a lumped element mass matrix $\mathcal{M}_e$ with arbitrary node masses m_i, the nodal and embedded point accelerations are

$$\ddot{\boldsymbol{x}}_e = \mathcal{M}_e^{-1} \boldsymbol{f}_e \tag{3.28}$$

$$\ddot{\boldsymbol{p}} = \mathcal{J}\ddot{\boldsymbol{x}}_e = \mathcal{J}\mathcal{M}_e^{-1}\boldsymbol{f}_e = \underbrace{\mathcal{J}\mathcal{M}_e^{-1}\mathcal{J}^T}_{\mathcal{M}_p^{-1}} \boldsymbol{f}_p = m_p \boldsymbol{f}_p \tag{3.29}$$

using the *effective mass* m_p

$$\mathcal{M}_p = \begin{bmatrix} m_p & 0 & 0 \\ 0 & m_p & 0 \\ 0 & 0 & m_p \end{bmatrix} \quad , \quad m_p = \frac{1}{\sum_{i=1}^{4} \frac{\xi_i^2}{m_i}} \tag{3.30}$$

that relates embedded point forces to accelerations and, from $\boldsymbol{j} = \Delta t \boldsymbol{f}$, can be used to relate impulses with velocity changes to obtain

$$\Delta\dot{\boldsymbol{p}} = \frac{\boldsymbol{j}_p}{m_p} \tag{3.31}$$

$$\Delta\dot{\boldsymbol{x}}_e = \mathcal{M}_e^{-1}\boldsymbol{j}_e = \mathcal{M}_e^{-1}\mathcal{J}^T\boldsymbol{j}_p = \mathcal{M}_e^{-1}\mathcal{J}^T m_p \Delta\dot{\boldsymbol{p}} \tag{3.32}$$

and from the last equation we can extract the simplified expressions for velocity and position change distribution on individual element nodes

$$\Delta\dot{\boldsymbol{x}}_i = \xi_i \frac{m_p}{m_i} \Delta\dot{\boldsymbol{p}} \tag{3.33}$$

$$\Delta\boldsymbol{x}_i = \xi_i \frac{m_p}{m_i} \Delta\boldsymbol{p} \tag{3.34}$$

where Equation (3.34) results from Equation (3.33) by converting displacements into time-integrated velocities $\Delta\boldsymbol{x}_e = \Delta t\dot{\boldsymbol{x}}_e$ and $\Delta\boldsymbol{p} = \Delta t\dot{\boldsymbol{p}}$.

A simpler distribution scheme that does not depend on node masses has also been proposed [WT08a], and results from the simplification of Equation (3.34) assuming uniform node masses $m_i = m$, that yields $\Delta\dot{\boldsymbol{x}}_i = \frac{\xi_i}{\sum_{j=1}^{4}\xi_j^2}\Delta\dot{\boldsymbol{p}}$. Although slightly more efficient, this distribution violates momentum conservation if node masses are actually different, a common fact for solid objects with heterogenous density or element sizes.

The previous expressions are trivially simplified when forces, impulses, position and velocity changes are applied directly on a single node $\boldsymbol{x}_i$.

3.4.2 Constraints

Constraints are algebraic relationships that must be fulfilled by the dynamics configuration simultaneously to the free body equations of motion

$$\mathcal{M}\ddot{\boldsymbol{q}} - \boldsymbol{f}(\boldsymbol{q},\dot{\boldsymbol{q}},t) = 0 \qquad \text{equations of motion} \tag{3.35}$$

$$g(\boldsymbol{q},\dot{\boldsymbol{q}},t) \geq 0 \qquad \text{inequality constraints} \tag{3.36}$$

$$h(\boldsymbol{q},\dot{\boldsymbol{q}},t) = 0 \qquad \text{equality constraints} \tag{3.37}$$

where the global mass matrix $\mathcal{M}$ and configuration vector $\boldsymbol{q}$ gather all mass and degrees of freedom (maximal coordinates) from all objects being simulated. Constraint expressions may include DOF derivatives, time, and external parameters. An exhaustive treatment and classification of constraints can be found in [GPS02]. For simplicity, we will focus on *geometric constraints* that only depend on the system configuration $\boldsymbol{q}$. Moreover, we shall only deal with unary equality constraints and binary inequality (contact) constraints defined on a pair of deformable objects A, B

$$h(\boldsymbol{x}_A) = 0 \qquad \text{unary inequality constraint} \tag{3.38}$$

$$g(\boldsymbol{x}_A,\boldsymbol{x}_B) \geq 0 \qquad \text{binary inequality constraint} \tag{3.39}$$

In computational physics and engineering simulations, constraints are known a priori and considered part of the problem to be solved or experiment to be performed. They are generally interpreted as persistent boundary conditions that can be used to derive specialized equations of motion. Symbolic embedding of constraint equations into the equations of motion yields a smaller set of unconstrained equations of motion defined on reduced coordinates [Bar96].

However, in the context of general purpose interactive simulation constraints are generally considered transient. While on-line symbolic constraint embedding is possible in some situations [GPHW05], the most flexible approach is to keep the equations of motion in maximal coordinates and represent the constraints as additional equations

that need to be solved. The ability to explicitly add/remove constraint equations through a user-interface or programatically is central to interactive simulation, and can be considered an *interaction*, as it modifies the equations of motion. Moreover, contact can be seen as a transient interaction between solid objects through contact constraints, that need to be automatically added/removed and solved.

Among existing constraint treatment schemes (see [Wri06] for an overview) the *penalty method* and the method of *Lagrange multipliers* are the most suitable for interactive simulation. We choose the latter, as it provides better visual accuracy and stability, at the expense of a more involved implementation. We will sidestep the computation of constraint forces, as we are not specifically interested in their value, and show instead how to modify the solution of Equation (3.3) to include the constraints directly in the iterative linear system solver, in a spirit similar to [BW98, AB03, Hau04]. In general, the constraint equations are only fulfilled weakly, at discrete times t^n, t^{n+1}, and may not hold during the interval Δt inbetween.

3.4.2.1 Equality constraints

We will first assume a single scalar equality constraint $h(\boldsymbol{x}) = 0$ for simplicity. By linearization around the current configuration $\boldsymbol{x}^n$ we obtain

$$h(\boldsymbol{x}^{n+1}) = h(\boldsymbol{x}^n) + \underbrace{\frac{\partial h}{\partial \boldsymbol{x}}\Big|_{x^n}}_{\mathcal{H}_{\boldsymbol{x}}} \Delta \boldsymbol{x} \tag{3.40}$$

$$\Delta h = \mathcal{H}_{\boldsymbol{x}} \Delta \boldsymbol{x} \tag{3.41}$$

assuming that constraints are currently satisfied $h(\boldsymbol{x}^n) = 0$, we want to enforce them at the next configuration $h(\boldsymbol{x}^{n+1}) = 0$ with $\Delta h = 0$, and obtain

$$\mathcal{H}_{\boldsymbol{x}} \Delta \boldsymbol{x} = 0 \tag{3.42}$$

this is, in fact, the *principle of virtual work* for constraint equations in discrete form, and states that constraint forces do no work or, equivalently, that admissible displacements are orthogonal to constraint gradients.

Recalling the Implicit Euler time-stepping scheme, each Newton-Raphson step k solves Equation (3.3) $\mathcal{A}_k^{n+1} \Delta \boldsymbol{x} = \boldsymbol{b}_k^{n+1}$ on the unknown displacements $\Delta \boldsymbol{x}$. We could use Equation (3.42) to remove a degree of freedom from Equation (3.3) following a cookbook elimination procedure, a method that extends to any number of constraint equations N_{eq} as long as they are compatible. This would be a suitable strategy if we used a direct method to solve Equation (3.3).

In an unconstrained iterative solver the solution to the system $\mathcal{A}\boldsymbol{z} = \boldsymbol{b}$ is successively approximated at each step m by the iterate $\boldsymbol{z}_m^-$. As proposed in [BW98], the iterative search directions can be *projected* to remove 1, 2 or 3 degrees of freedom from each

node $\boldsymbol{x}_i$ along forbidden directions $\boldsymbol{\eta}_j$ to guarantee admissible iterates $\boldsymbol{z}_m^+$, effectively reducing the solution space for $\boldsymbol{z}$. This method was proved to converge for Conjugate Gradients in [AB03] for constraints directly applied on nodes $\boldsymbol{x}_i$. The authors used 3×3 projection matrices $\mathcal{S}_i$ defined as

$$\mathcal{S}_i = \mathcal{I} - \sum_j \boldsymbol{\eta}_j \boldsymbol{\eta}_j^T \tag{3.43}$$

where per-node projections can be seen as assembled into a global projection matrix $\mathcal{S} = \text{diagonal}(\mathcal{S}_1 \dots \mathcal{S}_N)$, but are actually applied inside the Conjugate Gradients algorithm on a per-node basis to remove inadmissible changes to the iterates $\boldsymbol{z}_m^+$ along the directions $\boldsymbol{\eta}_j$. The linear system implicitly becomes $\mathcal{S}\mathcal{A}\boldsymbol{z} = \mathcal{S}\boldsymbol{b}$. In our case $\boldsymbol{z} = \Delta\boldsymbol{x}$, and, while obvious, it is interesting to notice that the global projection S that maps $\boldsymbol{z}_m^-$ into $\boldsymbol{z}_m^+$ is actually enforcing the principle of virtual work Equation (3.42) on the configuration displacements $\Delta\boldsymbol{x}$, and that the forbidden directions $\boldsymbol{\eta}_j$ are the transposed gradients $\mathcal{H}_{\boldsymbol{x}}$ of plane constraint equations $h_j(\boldsymbol{x}_i) = \boldsymbol{\eta}_j^T(\boldsymbol{x}_i - \boldsymbol{x}_i^0) = 0$ for an arbitrary position $\boldsymbol{x}_i^0$.

We can extend this approach to support constraints $h(\boldsymbol{p}(\boldsymbol{x}_e)) = 0$ on arbitrary embedded points and find a generalized projection matrix per element

$$\mathcal{S}_e = \mathcal{I} - \mathcal{H}_e^T \mathcal{H}_e \tag{3.44}$$

$$\mathcal{H}_e = \frac{\partial h}{\partial \boldsymbol{x}_e} = \frac{\partial h}{\partial \boldsymbol{p}} \frac{\partial \boldsymbol{p}}{\partial \boldsymbol{x}_e} = \mathcal{H}_{\boldsymbol{p}} \mathcal{J} \tag{3.45}$$

using the connector Jacobian $\mathcal{J}$. The projections $\mathcal{S}_e$ affects all nodes in the element and, as in the previous case, can be directly applied to the affected entries in $\Delta\boldsymbol{x}$, without being explicitly assembled into a global projection $\mathcal{S}$.

Using this approach equality constraints on nodes or embedded points can be efficiently added, removed and applied to the system at any timestep, as they only need to be stored in a list and sequentially applied as projections on the linear system solver iterates, with a total cost $\mathcal{O}(N_{\text{eq}})$ in both memory and CPU.

3.4.2.2 Contact constraints

A contact constraint on a pair of solids Ω_A, Ω_B in non-penetrating configuration can be defined by means of a gap function

$$g(\boldsymbol{p}_A, \boldsymbol{p}_B, \hat{\boldsymbol{n}}) = \hat{\boldsymbol{n}}^T(\boldsymbol{p}_A - \boldsymbol{p}_B) \geq 0 \tag{3.46}$$

where $\boldsymbol{p}_A, \boldsymbol{p}_B$ are a pair of contact points on the object surfaces Γ_A, Γ_B embedded in simplex elements, and $\hat{\boldsymbol{n}}$ is the contact normal that points from B to A by convention. These magnitudes are computed during the *contact determination* phase and will be

thoroughly discussed in Chapter 5. The gap function definition can be extended to return the penetration depth $g < 0$ when solids intersect.

The scalar constraint in Equation (3.46) states that objects must remain locally separated along the contact normal at the given points, but are free to slide tangentially or move away from each other. Thus, normal contact forces $\boldsymbol{f}_N = f_N \hat{\boldsymbol{n}}$ are not allowed to pull objects together, and must vanish on separation, which can be stated as the Signorini contact conditions, where $\perp$ represents complementarity $f_N g = 0$

$$0 \leq f_N \perp g \geq 0 \tag{3.47}$$

In addition to normal forces that avoid solid object interpenetration, realistic simulations must account for friction forces between solids in contact. Coulomb friction states the macroscopic relationship between normal $\boldsymbol{f}_N$ and tangential $\boldsymbol{f}_T$ contact forces as an upper bound on the latter, that defines a *friction cone*

$$\|\boldsymbol{f}_T\| \leq \mu f_N \tag{3.48}$$

An exact global solution of Equation (3.47) and Equation (3.48) for any number of contact constraints is hard due to the nonlinearity of Coulomb friction and the potential indeterminacy of contact forces in presence of friction [AP97]. To obtain a solution the problem can be reformulated at the velocity/impulse level

$$0 \leq j_N \perp \hat{\boldsymbol{n}}^T(\dot{\boldsymbol{p}}_A - \dot{\boldsymbol{p}}_B) \geq 0 \tag{3.49}$$

$$\|\boldsymbol{j}_T\| \leq \mu j_N \tag{3.50}$$

assuming constant normals $\hat{\boldsymbol{n}}$. Moreover, the nonlinear Coulomb friction cone can be discretized into a fixed number of tangential directions, resulting in a *friction pyramid*. The problem remains nonlinear due to the tight coupling between normal and friction impulses, that can be split by interleaving their computation. These common approximations result in a Linear Complementarity Problem (LCP) that can be efficiently solved using specifically optimized iterative solvers [DDKA06, OTSG09]. Hundreds of globally coupled contacts can be solved at interactive rates, but the computational cost remains too high for CPU-limited interactive applications in which only a fraction of the computational resources are available to physics simulation.

We relax the requirement of global coupling and use an approximate contact resolution method based in the application of local impulses on embedded points, in the line of [BFA02]. Using $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_A^T & \boldsymbol{x}_B^T \end{bmatrix}^T$, at each timestep $n \rightarrow n+1$ we perform contact determination to find all contact constraints. Then, at each Newton-Raphson step $k \rightarrow k+1$, we perform the following tasks

1. Solve $\mathcal{A}_k^{n+1} \Delta \boldsymbol{x}^- = \boldsymbol{b}_k^{n+1}$ applying equality constraints $\mathcal{H}_{\boldsymbol{x}} \Delta \boldsymbol{x}^- = 0$ by projection, ignoring any contact constraints.

2. Determine active contact set $\mathcal{C}$: Classify the contact as active if the relative displacement of the embedded points $\Delta \boldsymbol{p}_{AB}^{-} = \Delta \boldsymbol{p}_A(\Delta \boldsymbol{x}^-) - \Delta \boldsymbol{p}_B(\Delta \boldsymbol{x}^-)$ violates the separating condition $\hat{\boldsymbol{n}}^T \Delta \boldsymbol{p}_{AB}^{-} \geq 0$.

3. Enforce nonpenetration: For each active contact, we apply a displacement correction $\boldsymbol{d}_N = -\hat{\boldsymbol{n}}^T \Delta \boldsymbol{p}_{AB}^{-} \hat{\boldsymbol{n}}$ to cancel the relative displacement along $\hat{\boldsymbol{n}}$. The correction $\boldsymbol{d}_N$ needs to be distributed on Δx_A^- and Δx_B^- according to the inverse effective masses as

$$\boldsymbol{d}_{N_A} = w_A \boldsymbol{d}_N \qquad w_A = \frac{\frac{1}{m_A}}{\frac{1}{m_A} + \frac{1}{m_B}} \tag{3.51}$$

$$\boldsymbol{d}_{N_B} = -w_B \boldsymbol{d}_N \qquad w_B = 1 - w_A \tag{3.52}$$

and $\boldsymbol{d}_{N_A}$, $\boldsymbol{d}_{N_B}$ are applied on the embedded points using Equation (3.34). This correction can be seen as a pseudo-impulse that changes the candidate displacement $\Delta \boldsymbol{x}^-$ to obtain a corrected displacement $\Delta \boldsymbol{x}^+$ that avoids larger interpenetration at the next Newton-Raphson iterate $\boldsymbol{x}_{k+1}^{n+1}$.

4. Apply friction: After the normal pseudo-impulse is available we compute an analogous friction pseudo-impulse from direct application of Coulomb's law to displacements, instead of forces, with $\|\boldsymbol{d}_T\| \leq \mu \|\boldsymbol{d}_N\|$. The resulting friction pseudo-impulse is

$$\boldsymbol{d}_T = -\min(\mu \|\boldsymbol{d}_N\|, \hat{\boldsymbol{u}}^T \Delta \boldsymbol{p}_{AB}^{+}) \hat{\boldsymbol{u}} \tag{3.53}$$

where $\hat{\boldsymbol{u}} = \frac{\Delta \boldsymbol{p}_{AB}^{+}}{\|\Delta \boldsymbol{p}_{AB}^{+}\|}$ is the unitary displacement direction, and the upper bound avoids displacement direction reversal. As with the normal pseudo-impulse, $\boldsymbol{d}_T$ is distributed according to the weights w_A, w_B and applied to $\Delta \boldsymbol{x}^+$ using Equation (3.34), resulting in the final displacement $\Delta \boldsymbol{x}$.

5. Update iterates $\boldsymbol{x}_{k+1}^{n+1} = \boldsymbol{x}_k^{n+1} + \Delta \boldsymbol{x}$ and $\boldsymbol{v}_{k+1}^{n+1} = \frac{1}{\Delta t}(\boldsymbol{x}_k^{n+1} - \boldsymbol{x}^n)$

With a single Newton-Raphson iteration, $\Delta \boldsymbol{x}$ is the effective movement of the system $\boldsymbol{x}^n \to \boldsymbol{x}^{n+1}$ during the current timestep, and the applied normal and friction pseudo-impulses can be easily identified as the contact effect on such movement. For multiple Newton-Raphson iterations, however, $\Delta \boldsymbol{x}$ represents an incremental *correction* of the movement $\boldsymbol{x}^n \to \boldsymbol{x}^{n+1}$, and the corresponding pseudo-impulses should be identified as incremental corrections of the contact effects.

This locally-coupled impulse-based approach is simpler than state-of-the-art methods to solve globally coupled contacts [SMT08, OTSG09]. Applying pseudo-impulses directly on Newton-Raphson candidate displacements $\Delta \boldsymbol{x}$ is efficient, and correctly enforces the transition from tangential slip to sticking due to friction, while avoiding normal sticking in all cases. For soft objects a single Newton-Raphson iteration is usually enough, and results in a fast and robust method suitable for interactive simulation

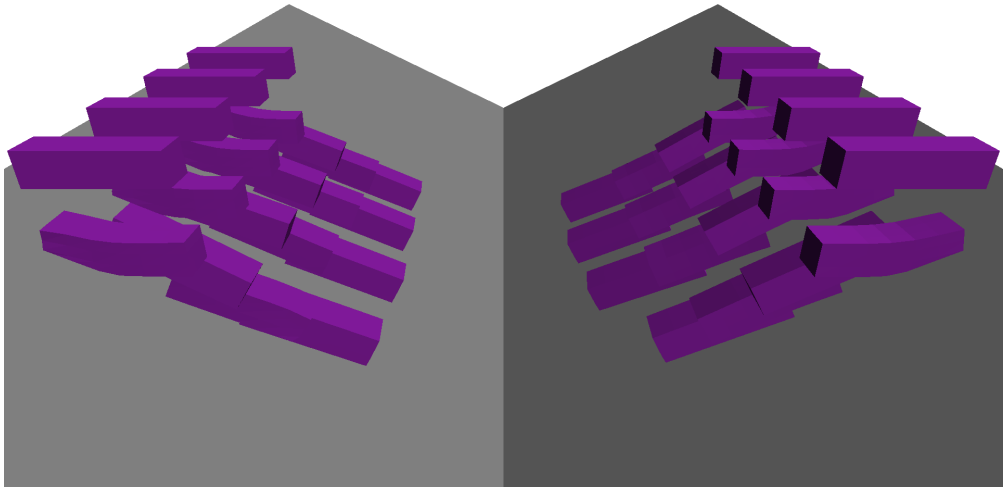

Figure 3.19: Contact reaction for Beam3D falling on a ramp from a height of 2m under $g = -9.8$, with friction coefficient $\mu = 0.75$. Left: $n_{\text{LS}} = 10$, Right: $n_{\text{LS}} = 20$. On both sides $n_{\text{NR}} = \{1, 2, 4, 8\}$ from front to back. Snapshots at $t = \{0, 0.5, 1.5, 3.75\}s$. Objects are static at $t = 3.75s$ due to friction.

with many contact points. For stiff objects the number of Newton-Raphson iterations should be increased to account for faster propagation of contact reaction over the whole material domain.

The effects of the number of linear system solver iterations n_{LS} and Newton-Raphson iterations n_{NR} on contact response can be seen in Figure 3.19. Increasing n_{LS} has a minor impact in contact response. Increasing n_{NR} results in faster impulse propagation that induces material vibration and allows the object to rattle on contact, with a slightly smaller apparent friction. Contact response for detailed surfaces with many embedded contact points can be seen in Figure 3.20. Additional contact response results for embedded detailed surfaces will be shown in Chapter 5.

	mass	E	ν	μ	Elements	Nodes
Armadillo10K	$0.65kg$	$1000Pa$	0.25	1	472	206
Dragon10K	$0.61kg$	$1000Pa$	0.25	0.75	408	173
Happy10K	$0.45kg$	$1000Pa$	0.25	0.5	361	142
Horse10K	$0.37kg$	$1000Pa$	0.25	0.5	287	146

Table 3.2: Models used in detailed contact test in Figure 3.20. All models have 10^4 embedded surface triangles.

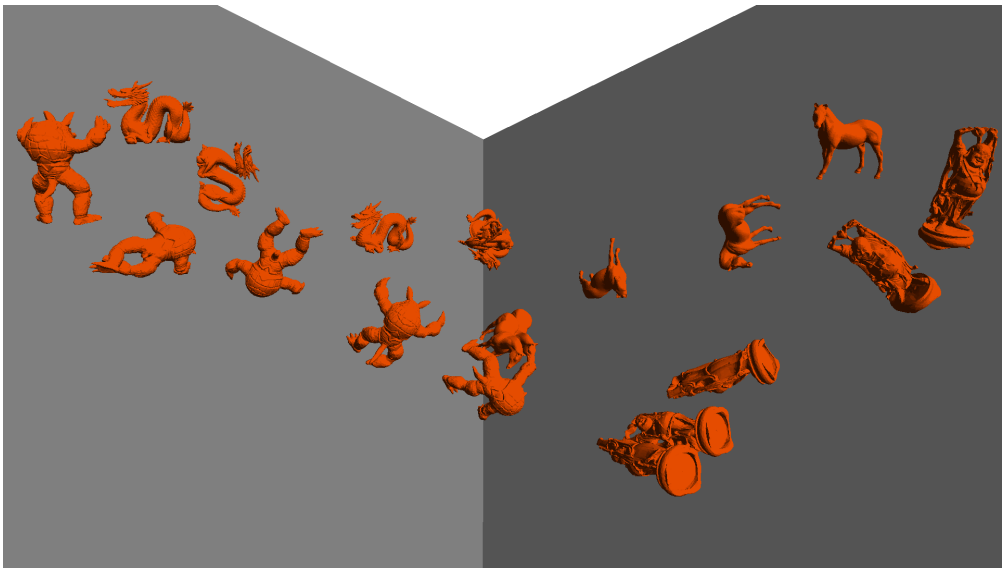

Figure 3.20: Contact reaction for embedded detailed surfaces with 10^4 triangles, falling on a ramp from a height of 1m under $g = -9.8$. Snapshots at $t = \{0, 1, 2.3, 3.7, 4.7\}s$ with $\{0, 77, 85, 118, 84\}$ embedded contact points. Models described in Table 3.2.

3.4.2.3 Constraint stabilization

Unfortunately, the assumption $h^n = h(\boldsymbol{p}(\boldsymbol{x}_e^n)) = 0$ rarely holds due to numerical errors in previous timesteps, caused by constraint linearization, finite precision and approximate solution of Equation (3.3). Numerical error may accumulate in successive timesteps as *constraint drift*, and requires a *constraint stabilization* method to avoid unbounded growth.

A possible strategy to avoid drift accumulation is *Baumagarte stabilization*, which adds a damped penalty force along the constraint gradient

$$\boldsymbol{f}_h = -(k_s h^n + k_d \dot{h}^n)\mathcal{H}_{\boldsymbol{p}}^T$$

where parameters k_s and k_d act as spring elasticity and damping coefficients. While simple and efficient, a major drawback of Baumagarte stabilization is that globally suitable coefficients are difficult to chose. Excessively large or small values may respectively induce instability or insufficient drift correction [ACPR94].

A more robust alternative is *post-stabilization*, that directly modifies the state of the system to enforce the constraint. Going back to Equation (3.41) with $h^n \neq 0$ and targeting for $h^{n+1} = 0$ we have

$$\mathcal{H}_e \Delta \boldsymbol{x}_e = \mathcal{H}_{\boldsymbol{p}} \Delta \boldsymbol{p} = -h^n \tag{3.54}$$

in order to revert constraint drift we can solve for the unknown embedded displacement using the left pseudoinverse $\mathcal{H}_{\boldsymbol{p}}^{+} = \mathcal{H}_{\boldsymbol{p}}^{T}(\mathcal{H}_{\boldsymbol{p}}\mathcal{H}_{\boldsymbol{p}}^{T})^{-1}$ [CP03]

$$\Delta \boldsymbol{p} = -\mathcal{H}_{\boldsymbol{p}}^{+} h^{n} \tag{3.55}$$

and consistently distribute it on element nodes using Equation (3.34). In practice, only a fraction of the correction displacement $k_r \Delta \boldsymbol{p}$ is added every timestep, using a relaxation coefficient $k_r \in (0, 1]$, to avoid introducing excessive discontiuity in $\boldsymbol{x}$. The computed displacement can be applied during the implicit step using *position alteration* as in [BW98], so that the time-stepping process is informed a priori of the stabilization displacement. Notice that for a plane constraint $h(\boldsymbol{p}(\boldsymbol{x}_e)) = \hat{\boldsymbol{n}}^T(\boldsymbol{p} - \boldsymbol{p}^0)$ with unit normal $\hat{\boldsymbol{n}}$ Equation (3.55) simplifies to $\Delta \boldsymbol{p} = -\hat{\boldsymbol{n}} h^n$, which can be computed very efficiently. Similar simplifications are possible for points constrained to move on a line or to remain at a fixed position.

Contact constraints often exhibit drift caused by discrete-time contact determination and varying contact normals in both time (due to global motion) and space (for non-planar surfaces), and need to be corrected to avoid visible interpenetration. Both stabilization schemes can be applied to contacts with $g^n < 0$. For Baumagarte, an action-reaction force pair $\boldsymbol{f}_A = -\boldsymbol{f}_B$ computed from Equation (3.4.2.3) needs to be applied to $\boldsymbol{p}_A$ and $\boldsymbol{p}_B$ in order to conserve momentum. For post-stabilization, the correction $\Delta \boldsymbol{p}_{AB} = -\hat{\boldsymbol{n}} g^n$ needs to be distributed on $\boldsymbol{p}_A$ and $\boldsymbol{p}_B$ according to the inverse effective masses as

$$\Delta \boldsymbol{p}_A = w_A \Delta \boldsymbol{p}_{AB} \qquad w_A = \frac{\frac{1}{m_A}}{\frac{1}{m_A} + \frac{1}{m_B}} \tag{3.56}$$

$$\Delta \boldsymbol{p}_B = -w_B \Delta \boldsymbol{p}_{AB} \qquad w_B = 1 - w_A \tag{3.57}$$

For interactive simulations we favor post-stabilization over Baumagarte, in order to achieve the same error correction fraction k_r regardless of object masses and external forces, which consistently results in less visible constraint violations for large timesteps. Both methods may add potential energy to the system. In case of Baumagarte, the elastic energy corresponding to an initially elongated zero-length multidimensional spring is added. For post-stabilization, the direct modification of node positions often adds elastic energy due to solid material compression, and may add or remove gravitational energy due to displacement along the gravity direction. In all cases the energy added grows with the timestep Δt, but does not cause long-term instability due to the numerical dissipation induced by Implicit Euler, that also grows with the timestep.

3.4.2.4 Constraint redundancy

Due to unrestricted interaction, redundant and incompatible constraints may be added to the system, either by direct creation of geometric constraints or automatic detec-

tion of multiple contacts on embedded points involving the same finite elements. In their presence, the augmented equation system formed by the Implicit Euler dynamics update in Equation (3.3) and the equality constraints in Equation (3.42) becomes singular. A solution may only exist in a least-squares sense, and direct methods must resort to singularity-tolerant solvers (eg. SVD) or regularization [Lac07], and remain too expensive for interactive simulation.

For contacts, redundancy becomes a serious issue in the simulation of deformable solids represented by highly detailed surfaces embedded in coarse simulation meshes, as the geometric detail is much higher than the dynamics detail, and often results in a large number of geometric contact points embedded in a few dynamics elements. Different strategies have been proposed to deal with redundant and competing contacts in constraint-based simulation: In [WT08a] the authors propose applying contact resolution to the single deepest embedded point in each tetrahedral element. In [AFC+10] volume-based contacts between detailed surfaces are detected and solved at the cells of a coarse grid, which avoids most redundancy. In rigid body simulation, redundant contacts are commonly reduced by clusterization or simplified to their convex hull, keeping only a subset of contacts that often include the deepest one. Some authors recommend avoiding constraint-based methods and fall back to penalty forces to cope with massive overdetermination in presence of lots of embedded point contacts [NKJF09].

In general, redundancy can be alleviated, but not completely eliminated. Fortunately, iterative solvers perform reasonably well for overconstrained linear systems, yet another reason for their popularity in interactive simulation. The displacement-level projection method detailed in Section 3.4.2.1 remains stable regardless of the number of active equality constraints. Overconstrained element nodes become transiently static, but do not cause instability or jitter. Similarly, the pseudo-impulsive contact response scheme proposed in Section 3.4.2.2, combined with the contact determination and reduction method proposed in Chapter 5 works well in practice and achieves visually pleasing results with a small CPU cost.

Drift cannot be completely avoided in presence of incompatible constraints, as no exact solution exists. Force-based Baumagarte stabilization can drive the system to a stable intermediate solution, but causes visible error even for compatible but redundant constraints. Sequential, Gauss-Seidel-like, post-stabilization leads to unstable configurations and visible jitter for incompatible constraints, and may over-correct compatible but redundant constraints. To avoid both problems, we average all aggregate corrections on each simulation node and apply them at once. We used this technique with successful results in Figure 3.19 and Figure 3.20. A globally coupled post-stabilization process could reconcile incompatible and redundant constraints in a fully mass-consistent way [CP03], but at a much higher CPU cost.

CHAPTER 4

Invertible Corotational FEM

The efficient and numerically robust simulation of deformable solids in presence of element degeneration will be discussed in this chapter, the state of the art will be reviewed and a new contribution will be presented in Section 4.4.

4.1 Introduction

Continuum mechanics does not generally deal with degenerate elements, as they would break the basic assumption that the volume of a material element cannot vanish, which represents a non-injective deformation map $\phi(\boldsymbol{X})$ that would greatly complicate mathematical treatment. The a priori assumption that any differential volume element $\mathrm{d}V$ will fulfill the non-degeneracy condition $J > 0$ is generally accepted. Computational modeling, however, is not limited (luckily, one could argue) by real world physics or a priori mathematical assumptions. After FEM material discretization of a D-dimensional solid domain into a set of N_e simplex elements and N_n nodes in $\mathbb{R}^D$, the unconstrained configuration space $\boldsymbol{x} \in \mathbb{R}^{ND}$ includes both regular and degenerate configurations.

Strictly enforcing non-degeneracy requires a reduced configuration space that satisfies the geometric constraint $\det(\mathcal{F}_e) > 0$ for all elements. This approach is used in [SKPSH13] to obtain locally injective mappings, but is computationally expensive and numerically difficult due to the large number of constraints and degrees of freedom involved.

An alternative method to guarantee non-degeneracy used in computational physics

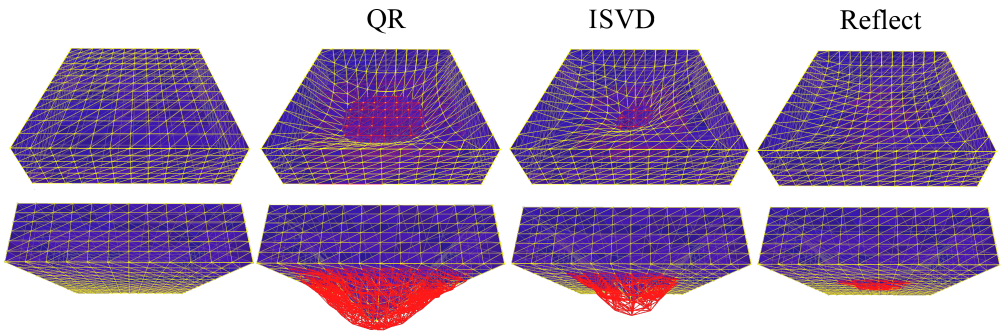

Figure 4.1: A box supported by all boundary faces except the top ones persistently collapses under gravity, with degenerate elements shown in red. Image from [CFS14].

[Wri06] are highly nonlinear elastic materials (eg.Neohookean) with an infinite energy barrier at the collapse limit $J = 0$. While conceptually elegant, this solution sidesteps the complexity of working with a reduced configuration space defined by geometric constraints at the expense of introducing singularities in the dynamics equations, that require arbitrarily small steps to be numerically integrated.

Unfortunately, strictly avoiding degeneration is not possible for interactive simulation, as unpredictable external causes may always produce it. Neither geometric constraints nor energy barriers are suitable solutions for unavoidable degeneracy treatment in computer graphics and interactive simulation. Moreover, their computational cost is too high, and severely limits the complexity of the scenes that can be simulated at interactive rates in non-dedicated hardware (eg. desktop PCs, gaming consoles, mobile phones, etc.). Thus, we will accept degeneration, and aim to maintain robustness and efficiency in the following scenarios:

- Slightly degenerate initial configurations due to modelling or precision errors.
- Transient degeneration due to collision, user interaction, or scripted animation.
- Persistent degeneration due to contact or kinematic constraints.

A robust and efficient method to *prevent*, *detect* and *correct* finite element degeneration is required.

4.2 Related work

The seminal article on invertible FEM [ITF04] proved that the nonlinear Green strain tensor $\mathcal{E} = \frac{1}{2}(\mathcal{F}^T\mathcal{F} - \mathcal{I})$ cannot detect element inversion due to its invariance to reflections, and proposed expressing elastic energy and stress in terms of $\mathcal{F}$ instead, and

introduced a method based in the Singular Value Decomposition that allows inversion handling for arbitrary constitutive material models. In such framework, collapse $|\det(\mathcal{F})| \leq \epsilon$ and inversion $\det(\mathcal{F}) < 0$ can be robustly detected.

While physically unrealistic due to its lack of rotational invariance, linear elasticity works as expected in presence of degenerate elements, as proved by [ST08]. On degeneration, the infinitesimal strain tensor $\epsilon_L = \frac{1}{2}(\mathcal{F}^T + \mathcal{F}) - \mathcal{I}$ successfully yields a larger elastic energy that produces correctly aligned forces that work to restore the element to the uninverted equilibrium configuration, as can be seen in the first panel in the top row of Figure 4.2.

In corotational elasticity, however, element degeneration interferes with the process of factoring the rotation $\mathcal{R}$ from the deformation gradient $\mathcal{F}$ by Polar Decomposition $\mathcal{F} = \mathcal{R}\mathcal{S}$, that fails on singular matrices with $\det(\mathcal{F}) = 0$ caused by collapsed elements, and returns a reflected rotation with $\det(\mathcal{R}) = -1$ on inverted elements with $\det(\mathcal{F}) < 0$, which results in elastic forces that further amplify inversion (second panel in the top row of Figure 4.2). Several solutions have been proposed in order to overcome the numerical and physical issues of polar decomposition on degenerate elements, listed in chronological order:

- ISVD: A generalization of polar decomposition for inverted configurations based in the Singular Value Decomposition was presented in [ITF04]. This method yields a symmetric $\mathcal{S}$, a proper rotation with $\det(\mathcal{R}) = 1$, and preserves rotational invariance in both global and material space, becoming the de-facto standard in computer animation.

- QR: A fast QR decomposition by Gram-Schmidt orthonormalization was used for both regular and degenerate configurations in [NPF05], at the expense of rotational invariace in material coordinates.

- CSVD: A modification of the ISVD approach that enforces time-consistent degeneration directions at the expense of continuity, introduced in [ST08].

- PD+QR: In [PO09], polar decomposition was used for regular elements, discontinuously switching to QR for nearly-collapsed or inverted elements with $\det(\mathcal{F}) < 0.06$.

- Project/Reflect: A modified polar decomposition method was proposed in our first publication [CFS14], and will be discussed in the next section.

An unrelated method for globally robust degeneration treatment based in the dynamic tetrahedralization of empty space has been recently presented in [MCKM15]. This method can globally untangle complex degeneration scenarios, but is computationally expensive and notably difficult to implement in 3D, and will not be furtherly discussed.

4.3 Project/Reflect

In the early stages of our implementation of a standard corotational linear FEM simulator we experienced an unexpected dynamical behaviour (a "glitch") when processing an isolated degenerate element with existing methods QR and ISVD: For some inverted configurations, the element would invert further before uninverting, often rotating counter-intuitively in the process. After discarding an error in our implementation, we found out that this unrealistic behaviours were structural, and also extended to CSVD. In [CFS14] we characterized the causes of such behaviour for each method on a single element, showed that they could produce macroscopic effects due to propagation to neighbour elements, and proposed the degeneracy treatment schemes Project and Reflect. The detailed discussion can be found in [CFS14] and will not be reproduced here. Instead, we will summarize the main results and propose an improved method in Section 4.4.

The main issues detected in previous degenerate element treatment methods are:

- **Discrete-time heuristic degeneration direction:** Any heuristic direction selection that only analyzes an instantaneous element configuration is bound to fail if large deformations are possible in a single simulation timestep. Enforcing time-consistency of the degeneration direction is not enough, and may consistently enforce a wrong recovery direction if the heuristic fails. Moreover, incorrect recovery directions may induce degeneration on neighbour elements. This issue affects all previous methods: QR, PD+QR, ISVD and CSVD.

- **Existence of critical points in inverted configurations:** Critical points in the computation of $\mathcal{R}$ as a function of $\mathcal{F}$ dramatically distort the inferred $\mathcal{R}$ around them, spanning all possible orientations. Corotational linear FEM intuitively *wraps* linear FEM forces around critical points which, in the inverted region, produce locally discontinuous and inversion-increasing forces that threaten both robustness and self-correction, and may result in equilibrium inverted configurations. This issue affects QR and ISVD, as shown in the the corresponding panels in Figure 4.2, where the critical points act as vector field sources.

- **Discontinuity of $\mathcal{R}$:**. Any discontinuity in the computation of $\mathcal{R}$ is bound to cause discontinuous forces that result in instable dynamics and visible jitter, and may severely interfere with Newton-Raphson convergence. This issue affects CSVD (Figure 4.2) and PD+QR (not explicitly pictured).

Our main contributions were:

- Computing a chronologically-correct, time-consistent and continuous degeneration direction $\hat{\boldsymbol{d}}_c$.

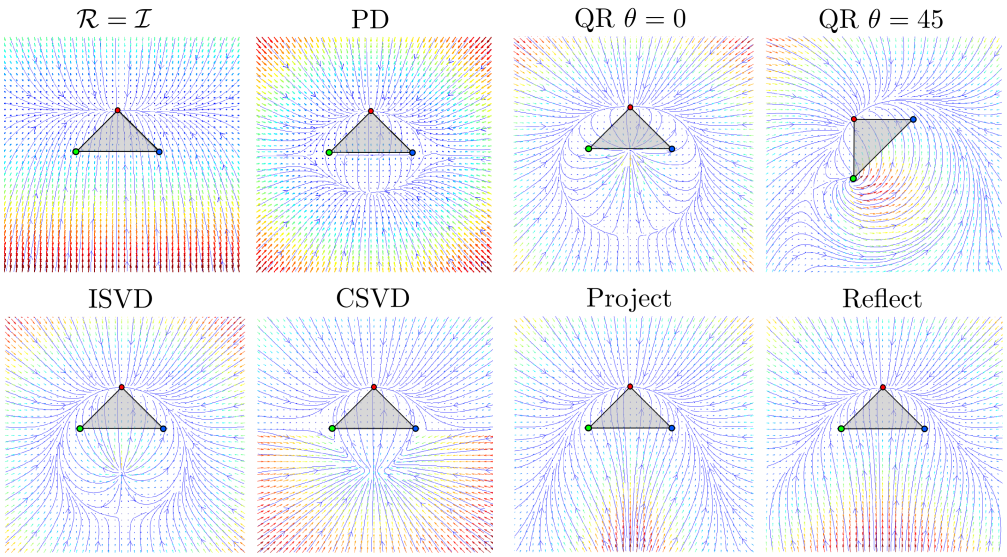

Figure 4.2: Force fields experienced by node $\boldsymbol{x}_1$ (red) in a $[-3, 3]^2$ region of the plane when $\boldsymbol{x}_2$ (green) and $\boldsymbol{x}_3$ (blue) are fixed, for corotational linear FEM with rotation extraction methods in previous works: Linear elasticity with no rotation ($\mathcal{R} = \mathcal{I}$), Polar Decomposition [MG04], QR with unrotated ($\theta = 0$) and rotated ($\theta = 45$) reference configurations[NPF05], Invertible SVD [ITF04], Coherent SVD [ST08], Project and Reflect [CFS14]. Field arrows show force direction and magnitude, with warmer colors representing stronger forces. Streamlines represent trajectories of inertialess particles in the force field. Material parameters $E = 1000Pa$, $\nu = 0.35$.

- Computing a continuous rotation $\bar{\mathcal{R}}$ that matches $\mathcal{R}$ from polar decomposition in regular configurations, but avoids critical points in degenerate configurations.

The intuition behind the proposed method was to avoid rotation beyond collapse, an idea that will be detailed in Section 4.4. A good property of the method is that it only modifies elastic forces on degenerate elements to encourage robust and fast recovery along $\hat{\boldsymbol{d}}_c$ while respecting their constitutive material model, without resorting to additional mechanisms such as artificial springs, additional pressure terms or hard constraints, that would indirectly change the material properties and increase the computational cost. A visual summary of the results in [CFS14] can be found in Figure 4.1 and Figure 4.2.

As a limitation, the Project/Reflect methods were derived from a purely geometric perspective and resulted in a continuous but non-differentiable $\bar{\mathcal{R}}$ that did not allow stable fully-implicit integration, as $\delta\bar{\mathcal{R}}$ was discontinuous for any differential displacement $\delta\boldsymbol{x}$ across a collapse configuration.

4.4 Degeneration-Aware Polar Decomposition (DAPD)

In this section we introduce DAPD, a differentiable generalization of the continuous-time approach presented in [CFS14]. DAPD computes an undegenerate rotation $\bar{\mathcal{R}}$ and, optionally, its differential $\delta\bar{\mathcal{R}}$, that are required during implicit integration to obtain the elastic forces and their differentials, respectively. The whole process is detailed in the following sections. The discussion will be kept dimension-independent, using the parameter D, except when differences between 2D triangles and 3D tetrahedrons exist.

4.4.1 Degeneration detection

Our degeneration detection method is inspired in continuous collision detection. During a simulation step $[t_0, t_1]$, an initially undegenerate simplex element with $\det(\mathcal{F}(t_0)) > 0$ can only become degenerate at t_1 with $\det(\mathcal{F}(t_1)) \leq 0$ if it has collapsed an odd number of times, with the first collapse $\det(\mathcal{F}(t_c)) = 0$ occurring at $t_c \in (t_0, t_1]$. Elements that have collapsed an even number of times during a single interval are considered self-corrected and not treated as degenerate.

Once an element is considered degenerate, we interpolate node trajectories linearly $\boldsymbol{x}_i(t) = \boldsymbol{x}_i(t_0) + \frac{t-t_0}{t_1-t_0}\big(\boldsymbol{x}_i(t_1) - \boldsymbol{x}_i(t_0)\big)$ and solve the polynomial equation $\det(\mathcal{F}(t)) = \alpha$, which is quadratic for triangles and cubic for tetrahedrons. We use a positive threshold α instead of strictly 0 to treat flattened elements below a fraction α of their original volume as effectively collapsed, in order to improve numerical robustness.

The element configuration at the time of collapse t_c is then analysed as in [CFS14], and the pair of geometric features that became nearly coincident is identified. We save this witness feature pair (A, B) for each degenerate element until it recovers, and use it every timestep to compute a chronologically-correct and time-consistent unitary *degeneration direction* $\hat{\boldsymbol{d}}_c = \frac{\boldsymbol{u}}{\|\boldsymbol{u}\|}$ with sign chosen by convention to point from A to B. For tetrahedrons in 3D only vertex-face and edge-edge pairs are considered essential. A vertex-face pair (V_i, F_{jkl}) uses the face normal direction $\boldsymbol{u}^{(V_i, F_{jkl})} = \boldsymbol{x}_{jk} \times \boldsymbol{x}_{jl}$ and an edge-edge pair (E_{ij}, E_{kl}) uses the direction normal to both edges $\boldsymbol{u}^{(E_{ij}, E_{kl})} = \boldsymbol{x}_{ij} \times \boldsymbol{x}_{kl}$. For triangles in 2D only the vertex-edge (V_i, E_{jk}) case is essential, with $\boldsymbol{u}^{(V_i, E_{jk})} = \boldsymbol{x}_{jk}^{\perp}$, perpendicular to E_{jk}. Non-essential collapse feature pairs represent geometric coincidences (eg. (V_i, V_j), (V_i, E_{jk}), (F_i, E_{jk})) that, while possible, can always be promoted to an essential case that yields the *same* degeneration direction $\hat{\boldsymbol{d}}_c$ at t_c.

4.4.2 Rotation $\bar{\mathcal{R}}$

Given a degenerate element with $\det(\mathcal{F}) < \alpha$ we use its persistent collapse feature pair (A, B) to compute a *virtual* undegenerate configuration $\bar{\mathcal{D}}_s$ and perform standard Polar Decomposition on the modified deformation matrix $\bar{\mathcal{F}} = \bar{\mathcal{D}}_s \mathcal{D}_m^{-1}$ to extract a rotation

$\bar{\mathcal{R}}$ that, intuitively, does not allow the element to rotate beyond collapse. Further, we guarantee that $\det(\bar{\mathcal{F}}) > \epsilon$, for a positive ϵ large enough to avoid any numerical issues in the Polar Decomposition.

The undegenerate configuration $\bar{\mathcal{D}}_s$ is computed differently in the vertex-face and edge-edge collapse cases. In the vertex-face case (V_1, F_{234}) we only displace the node $\boldsymbol{x}_1$ to a virtual position $\bar{\boldsymbol{x}}_1$, while in the edge-edge case (E_{12}, E_{34}) we apply the same displacement to both $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. In the vertex-edge 2D case we only displace $\boldsymbol{x}_1$

$$\bar{\boldsymbol{x}}_i = \boldsymbol{x}_i + \lambda \hat{\boldsymbol{d}}_c \tag{4.1}$$

$$\bar{\mathcal{D}}_s^{(V_1,F_{234})} = \begin{bmatrix} \boldsymbol{x}_2 - \bar{\boldsymbol{x}}_1 & \boldsymbol{x}_3 - \bar{\boldsymbol{x}}_1 & \boldsymbol{x}_4 - \bar{\boldsymbol{x}}_1 \end{bmatrix} \tag{4.2}$$

$$\bar{\mathcal{D}}_s^{(E_{12},E_{34})} = \begin{bmatrix} \bar{\boldsymbol{x}}_2 - \bar{\boldsymbol{x}}_1 & \boldsymbol{x}_3 - \bar{\boldsymbol{x}}_1 & \boldsymbol{x}_4 - \bar{\boldsymbol{x}}_1 \end{bmatrix} \tag{4.3}$$

$$\bar{\mathcal{D}}_s^{(V_1,E_{23})} = \begin{bmatrix} \boldsymbol{x}_2 - \bar{\boldsymbol{x}}_1 & \boldsymbol{x}_3 - \bar{\boldsymbol{x}}_1 \end{bmatrix} \tag{4.4}$$

Expressions for different feature pairs can be trivially found by index permutation.

The direction $\hat{\boldsymbol{d}}_c$ depends on the collapse feature pair type as detailed in the previous section. We compute the degeneration depth w along $\hat{\boldsymbol{d}}_c$ using one node on each feature in (A, B) as

$$w = (\boldsymbol{x}_1 - \boldsymbol{x}_3)^T \hat{\boldsymbol{d}}_c \tag{4.5}$$

The degeneration threshold h is computed so that $\det(\bar{\mathcal{F}}) = \alpha$ when $w = h$. The actual expression for h depends on the collapse feature pair type:

$$h^{(A,B)} = \alpha \det(\mathcal{D}_m) / \|\boldsymbol{u}^{(A,B)}\| \tag{4.6}$$

where $\|\boldsymbol{u}^{(A,B)}\|$ is the appropriate area (length in 2D) term that relates the tetrahedron volume (triangle area in 2D) with its height along $\hat{\boldsymbol{d}}_c$. Finally, the displacement length λ that guarantees $\det(\bar{\mathcal{F}}) > \epsilon$ is computed as:

$$\lambda(w,h) = \begin{cases} 0 & \text{if } w \geq h \\ \lambda_C & \text{if } 0 \leq w < h \\ \lambda_I & \text{if } w < 0 \end{cases} \tag{4.7}$$

with

$$s = -\frac{w}{h} \tag{4.8}$$

$$r = 1 + s \tag{4.9}$$

$$\lambda_C = \beta h (2r^2 - r^3) \tag{4.10}$$

$$\lambda_I = \beta h r \tag{4.11}$$

where the auxiliary variables r and s are used to define a cubic spline interpolation curve in the middle region $0 \leq w < h$ of the piecewise function λ in Equation (4.7) that

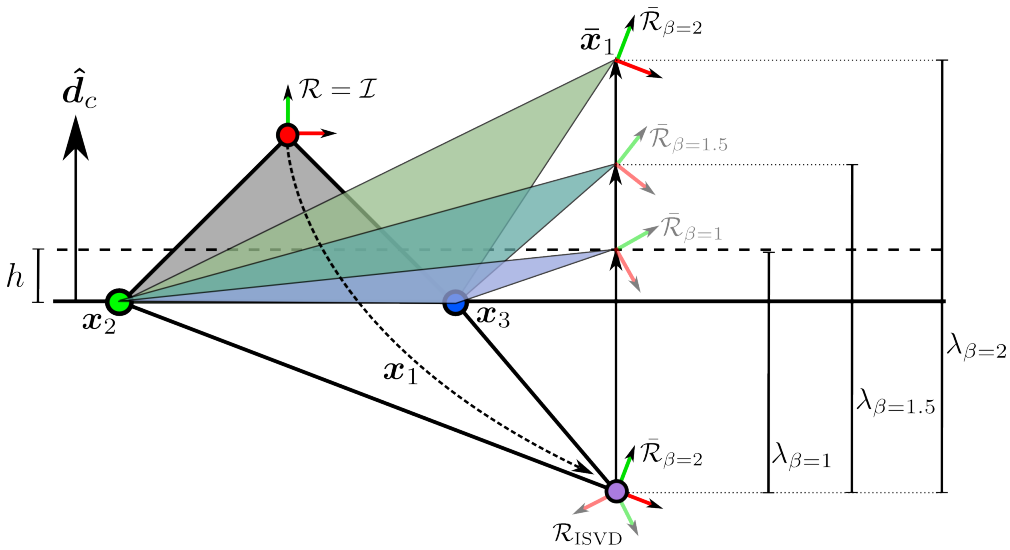

Figure 4.3: Schematics of the DAPD process for different values of β. In the picture the reference triangle element becomes degenerate when V_1 moves to the violet point across the opposite edge E_{23}. The dotted horizontal line shows the collapse threshold h that depends on the volume change fraction threshold α. Any position $\boldsymbol{x}_1$ below this threshold is considered degenerate. The DAPD algorithm displaces the node to a virtual position $\bar{\boldsymbol{x}}_1$ above this threshold, and computes $\bar{\mathcal{R}}$ using the resulting triangle $(\bar{\boldsymbol{x}}_1, \boldsymbol{x}_2, \boldsymbol{x}_3)$, shown for 3 different values of $\beta = 1, 1.5, 2$. As an example, the rotation $\bar{\mathcal{R}}_{\beta=2}$ is selected and assigned to the actual degenerate triangle. By construction, DAPD never generates a rotation $\bar{\mathcal{R}}$ that corresponds to a degenerate triangle, and therefore avoids the "rotation past collapse" inconsistency that affects ISVD, seen in $\mathcal{R}_{\text{ISVD}}$. In the diagram each possible rotation is drawn at the corresponding position of triangle vertex V_1 to emphasize the dependency.

smoothly connects the undegenerate $w \geq h$ and inverted $w < 0$ regions. The parameter $\beta \geq 1$ generates a continuous range of correction displacements. Its schematic effect can be seen in Figure 4.3. For $\beta = 1$ and $\beta = 2$, the results of DAPD approximately match the Project and Reflect methods in [CFS14], respectively. As β grows, the rotations and forces on the inverted nodes become more coherently aligned (Figure 4.4).

We can now compute $\bar{\mathcal{F}} = \bar{\mathcal{D}}_s \mathcal{D}_m^{-1}$ from $\bar{\mathcal{D}}_s$ and perform polar decomposition to obtain $\bar{\mathcal{R}}$. Finally, we compute the local deformation $\hat{\mathcal{S}} = \bar{\mathcal{R}}^T \mathcal{F}$ that will be required to obtain the stress $\mathcal{P}$ as detailed in the following subsections. Pseudocode for the whole process is shown in Algorithm 1.

4.4.3 Rotation differential $\delta\bar{\mathcal{R}}$

As detailed in Section 3.3.3, implicit integration with exact stress differentials $\delta\mathcal{P}$ requires the rotation differentials $\delta\bar{\mathcal{R}}$. From the given node displacements $\delta\boldsymbol{x}_i$, we com-

Algorithm 1 Compute $\bar{\mathcal{F}}$,$\bar{\mathcal{R}}$,$\bar{\mathcal{S}}$,$\hat{\mathcal{S}}$

procedure ComputeFRS($\boldsymbol{x}_1 \dots \boldsymbol{x}_{D+1}$, A, B)
  $\mathcal{D}_s \leftarrow \begin{bmatrix}\boldsymbol{x}_2 - \boldsymbol{x}_1 & \dots & \boldsymbol{x}_{D+1} - \boldsymbol{x}_1\end{bmatrix}$
  $\mathcal{F} \leftarrow \mathcal{D}_s \mathcal{D}_m^{-1}$
  if $\det(\mathcal{F}) \leq \alpha$ **then** ▷ degenerate
    $\hat{\boldsymbol{d}}_c \leftarrow \boldsymbol{u}^{(A,B)} / \|\boldsymbol{u}^{(A,B)}\|$ ▷ case-specific direction
    for $i \leftarrow 1 \dots D+1$ **do** ▷ displace feature A
      if $i \in A$ **then**
        $\bar{\boldsymbol{x}}_i \leftarrow \boldsymbol{x}_i + \lambda^{(A,B)} \hat{\boldsymbol{d}}_c$ ▷ case-specific displacement
      else
        $\bar{\boldsymbol{x}}_i \leftarrow \boldsymbol{x}_i$
      end if
    end for
    $\bar{\mathcal{D}}_s \leftarrow \begin{bmatrix}\bar{\boldsymbol{x}}_2 - \bar{\boldsymbol{x}}_1 & \dots & \bar{\boldsymbol{x}}_{D+1} - \bar{\boldsymbol{x}}_1\end{bmatrix}$
    $\bar{\mathcal{F}} \leftarrow \bar{\mathcal{D}}_s \mathcal{D}_m^{-1}$
  else ▷ undegenerate
    $\bar{\mathcal{F}} \leftarrow \mathcal{F}$
  end if
  $\bar{\mathcal{R}} \leftarrow$ PolarDecomposition($\bar{\mathcal{F}}$) ▷ standard PD
  $\bar{\mathcal{S}} \leftarrow \bar{\mathcal{R}}^T \bar{\mathcal{F}}$
  $\hat{\mathcal{S}} \leftarrow \bar{\mathcal{R}}^T \mathcal{F}$
  return $\{\bar{\mathcal{F}},\bar{\mathcal{R}},\bar{\mathcal{S}},\hat{\mathcal{S}}\}$
end procedure

pute $\delta\bar{\mathcal{D}}_s$ as

$$\delta\bar{\boldsymbol{x}}_i = \delta\boldsymbol{x}_i + \delta\lambda\hat{\boldsymbol{d}}_c + \lambda\delta\hat{\boldsymbol{d}}_c \tag{4.12}$$

$$\delta\bar{\mathcal{D}}_s^{(V_1,F_{234})} = \begin{bmatrix}\delta\boldsymbol{x}_2 - \delta\bar{\boldsymbol{x}}_1 & \delta\boldsymbol{x}_3 - \delta\bar{\boldsymbol{x}}_1 & \delta\boldsymbol{x}_4 - \delta\bar{\boldsymbol{x}}_1\end{bmatrix} \tag{4.13}$$

$$\delta\bar{\mathcal{D}}_s^{(E_{12},E_{34})} = \begin{bmatrix}\delta\bar{\boldsymbol{x}}_2 - \delta\bar{\boldsymbol{x}}_1 & \delta\boldsymbol{x}_3 - \delta\bar{\boldsymbol{x}}_1 & \delta\boldsymbol{x}_4 - \delta\bar{\boldsymbol{x}}_1\end{bmatrix} \tag{4.14}$$

$$\delta\bar{\mathcal{D}}_s^{(V_1,E_{23})} = \begin{bmatrix}\delta\boldsymbol{x}_2 - \delta\bar{\boldsymbol{x}}_1 & \delta\boldsymbol{x}_3 - \delta\bar{\boldsymbol{x}}_1\end{bmatrix} \tag{4.15}$$

The differential of the piecewise function $\lambda(w,h)$ is

$$\delta\lambda(w,h) = \begin{cases} 0 & \text{if } w \geq h \\ \frac{\partial\lambda_C}{\partial w}\delta w + \frac{\partial\lambda_C}{\partial h}\delta h & \text{if } 0 \leq w < h \\ \frac{\partial\lambda_I}{\partial w}\delta w + \frac{\partial\lambda_I}{\partial h}\delta h & \text{if } w < 0 \end{cases} \tag{4.16}$$

with

$$\frac{\partial\lambda_C}{\partial w} = -\beta(4r - 3r^2) \qquad \frac{\partial\lambda_I}{\partial w} = -\beta$$
$$\frac{\partial\lambda_C}{\partial h} = \beta(1 + s^2 + 2s^3) \qquad \frac{\partial\lambda_I}{\partial h} = \beta$$

and

$$\delta w = \delta\boldsymbol{x}_{31}^T\hat{\boldsymbol{d}}_c + \boldsymbol{x}_{31}^T\delta\hat{\boldsymbol{d}}_c \tag{4.17}$$

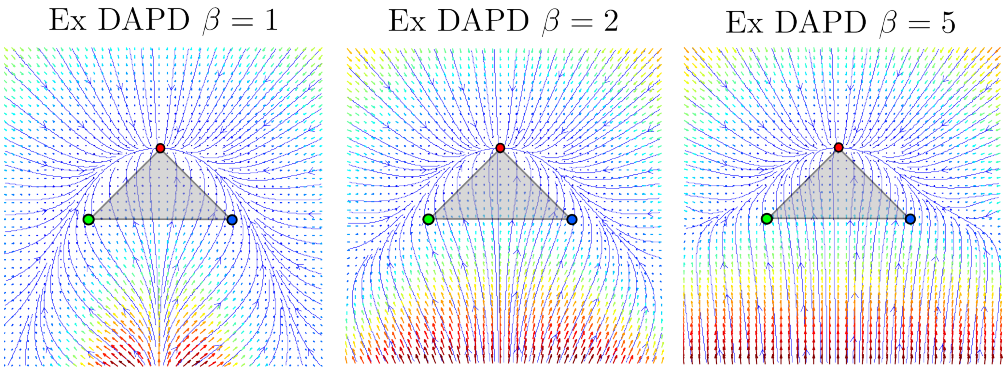

Figure 4.4: Exact force fields experienced by node $\boldsymbol{x}_1$ (red) in a $[-3, 3]^2$ region of the plane when $\boldsymbol{x}_2$ (green) and $\boldsymbol{x}_3$ (blue) are fixed, for corotational linear FEM with DAPD rotation extraction for different values of β. Degeneracy threshold $\alpha = 0.1$, material parameters $E = 1000Pa$, $\nu = 0.35$.

The terms δh and $\delta\hat{\boldsymbol{d}}_c$ can be computed generically for any feature pair (A, B) as

$$\delta h = -\frac{\alpha \det(\mathcal{D}_m)\boldsymbol{u}^T\delta\boldsymbol{u}}{\|\boldsymbol{u}\|^3} \tag{4.18}$$

$$\delta\hat{\boldsymbol{d}}_c = \frac{\delta\boldsymbol{u}\|\boldsymbol{u}\|^2 - \boldsymbol{u}[\boldsymbol{u}^T\delta\boldsymbol{u}]}{\|\boldsymbol{u}\|^3} \tag{4.19}$$

using the case-specific term $\boldsymbol{u}^{(A,B)}$ and its differential

$$\delta\boldsymbol{u}^{(V_1,F_{234})} = x_{23} \times \delta x_{24} + \delta\boldsymbol{x}_{23} \times x_{24} \tag{4.20}$$

$$\delta\boldsymbol{u}^{(E_{12},E_{34})} = x_{12} \times \delta x_{34} + \delta\boldsymbol{x}_{12} \times x_{34} \tag{4.21}$$

$$\delta\boldsymbol{u}^{(V_1,E_{23})} = \delta x_{23}^{\perp} \tag{4.22}$$

We can now compute $\delta\bar{\mathcal{F}} = \delta\bar{\mathcal{D}}_s\mathcal{D}_m^{-1}$ and $\bar{\mathcal{S}} = \bar{\mathcal{R}}^T\bar{\mathcal{F}}$, and use Equation (3.13) to obtain the $\delta\bar{\mathcal{R}}$ required by exact $\delta\mathcal{P}$, as detailed in the following subsections.

4.4.4 Continuity and differentiability

The Polar Decomposition is continuous and differentiable for undegenerate configurations. For DAPD, it can be shown from Eq.4.7 that $\lambda(w, h)$ is continuous and differentiable in w and h as long as the denominator term $\|\boldsymbol{u}\|$ is nonzero, as all piecewise terms and their derivatives match at the transition cases $w = h$ and $w = 0$. The continuity and differentiability of $\lambda(w, h)$ extends to $\bar{\mathcal{D}}_s$, $\bar{\mathcal{F}}$, $\bar{\mathcal{R}}$, $\hat{\mathcal{S}}$, $\bar{\mathcal{P}}$ and ultimately to $\boldsymbol{f}_e$, as DAPD always evaluates the Polar Decomposition and its differential in undegenerate configurations. The denominator term $\|\boldsymbol{u}\|$ may only become zero for a tetrahedron in the following cases:

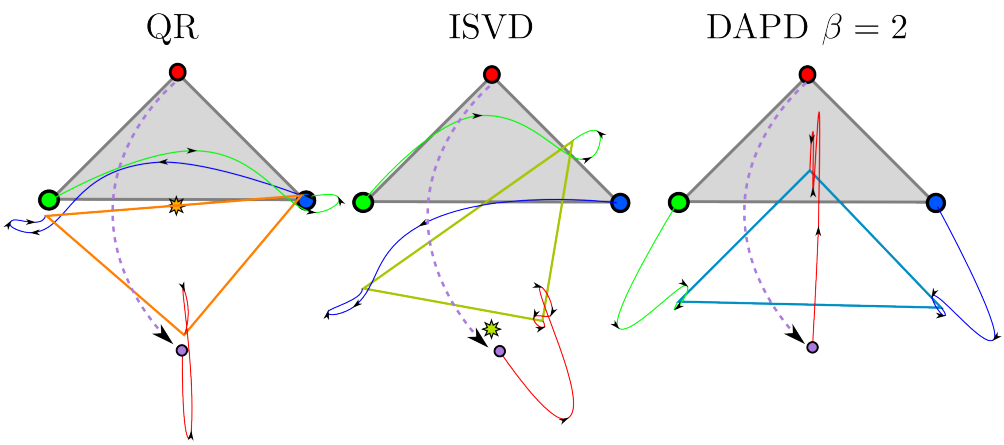

Figure 4.5: A triangle element in reference configuration (light grey) is inverted by moving the red node across its opposite edge along the dotted violet line. The simulation is then started and all 3 node trajectories plotted in their respective colours, until equilibrium. Left and middle panels show that the element resolves inversion by performing a large rotation in QR and ISVD. The DAPD method in the rightmost panel avoids such rotation, using $\alpha = 0.1$ and $\beta = 3$. Critical points of QR and ISVD shown as stars. Material parameters $E = 1000$, $\nu = 0.35$, damping ratio $= 0.25$.

- Face F_{234} collapses to a segment or point in a (V_1, F_{234}) pair.
- Edges E_{12} or E_{34} collapse or become parallel in a (E_{12}, E_{34}) pair.

In the 2D case, the denominator can only be zero if the edge E_{23} collapses in a (V_1, E_{23}) pair. All mentioned situations require an element to collapse in a specific way while already degenerate. We can instantaneously define $\bar{\mathcal{R}} = \mathcal{I}$ and $\delta\bar{\mathcal{R}} = 0$ to perturbate the system away from such extremely degenerate configurations. Continuity can be improved if we store $\bar{\mathcal{R}}$ from the previous timestep and reuse it until $\boldsymbol{u}$ is well defined again. This is still discontinuous, but will only affect the simulation shortly, typically for a single timestep. Similar discontinuous rotation definitions were used in [TSIF05] and [PO09] for highly degenerate configurations.

4.4.5 Computing $\mathcal{P}$ and $\delta\mathcal{P}$

We will reinterpret the modification of the polar decomposition rotation $\mathcal{R}$ into $\bar{\mathcal{R}}$ for degenerate configurations as a modification of the standard corotational strain tensor ϵ_C from Section 2.1.4.2 into a *degeneration-aware corotational strain tensor*. This will provide additional insight into the behaviour of degenerate elements. We define the

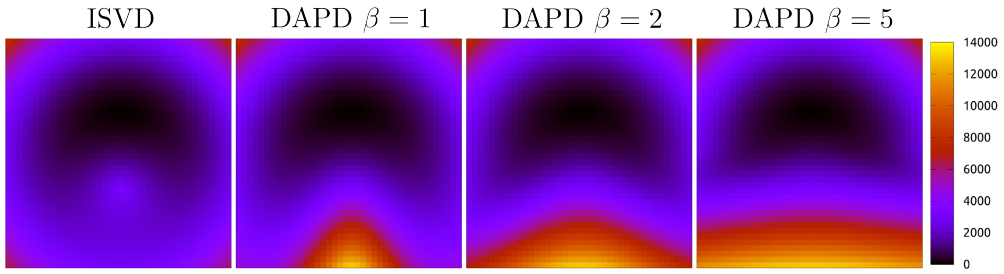

Figure 4.6: Elastic energy field in a $[-3, 3]^2$ region for the example triangle element with material parameters $E = 1000$, $\nu = 0.35$ and DAPD $\alpha = 0.1$. Notice the larger energy for degenerate configurations in DAPD, that grows with β, and the local maximum in ISVD, that cause inversion-increasing forces. All fields are equal in the undegenerate region $y_1 > h$.

degeneration-aware strain and elastic energy as

$$\epsilon_D = \frac{1}{2}(\hat{\mathcal{S}}^T + \hat{\mathcal{S}}) - \mathcal{I} \tag{4.23}$$

$$\Psi_D = \mu \|\epsilon_D\|_F^2 + \frac{\lambda}{2} \operatorname{tr}^2(\epsilon_D) \tag{4.24}$$

where $\hat{\mathcal{S}} = \bar{\mathcal{R}}^T \mathcal{F}$ is the deformation that remains after removing the modified rotation $\bar{\mathcal{R}}$ from the total deformation $\mathcal{F}$. The elastic energy field can be seen in Figure 4.6. Its intensity grows steadily when triangle configurations become more inverted, thanks to the modified strain measure, as opposed to the standard corotational energy Ψ_C using ISVD, that shows a local maximum in the inverted region, responsible of the inversion-increasing local forces described in [CFS14].

The strain tensor ϵ_D is always symmetric, but our modified $\bar{\mathcal{R}}$ does not yield a symmetric $\hat{\mathcal{S}}$ for degenerate elements in general position. Due to the potential asymmetry of $\hat{\mathcal{S}}$, the stress $\mathcal{P}_C$ and stress differential $\delta\mathcal{P}_C$ expressions from corotational linear FEM are not directly applicable. Their step-by-step derivation can be found in the Technical Notes associated to [MZS+11a], and results in the compact expressions in Equation (3.15) and Equation (3.16) thanks to the *strategic cancellation* of several terms due to the symmetry of $\mathcal{S}$ from the standard polar decomposition $\mathcal{F} = \mathcal{R}\mathcal{S}$. We recall the expressions here

$$\mathcal{P}_C = \mathcal{R}[2\mu\epsilon_C + \lambda \operatorname{tr}(\epsilon_C)\mathcal{I}] \qquad \text{Equation (3.15)}$$

$$\delta\mathcal{P}_C = 2\mu\delta\mathcal{F} + \lambda \operatorname{tr}(\mathcal{R}^T\delta\mathcal{F})\mathcal{R} + [\lambda \operatorname{tr}(\epsilon_C) - 2\mu]\delta\mathcal{R} \qquad \text{Equation (3.16)}$$

Such cancellation does exist for a non-symmetric $\hat{\mathcal{S}}$, which will complicate the derivation of $\mathcal{P}_D$ and $\delta\mathcal{P}_D$ from Ψ_D. Details can be found in Appendix A.1. The resulting

expressions using the new ϵ_D are

$$\mathcal{P}_D = \underbrace{\bar{\mathcal{R}}[2\mu\epsilon_D + \lambda\,\mathrm{tr}(\epsilon_D)\mathcal{I}]}_{\hat{\mathcal{P}}_D} + \check{\mathcal{P}}_D \tag{4.25}$$

$$\delta\mathcal{P}_D = \underbrace{2\mu\delta\bar{\mathcal{F}} + \lambda\,\mathrm{tr}(\bar{\mathcal{R}}^T\delta\bar{\mathcal{F}})\bar{\mathcal{R}} + [\lambda\,\mathrm{tr}(\epsilon_D) - 2\mu]\delta\bar{\mathcal{R}}}_{\delta\hat{\mathcal{P}}_D} + \delta\check{\mathcal{P}}_D \tag{4.26}$$

where the terms $\check{\mathcal{P}}_D$ and $\delta\check{\mathcal{P}}_D$ appear due to non-symmetry of $\hat{\mathcal{S}}$, and only vanish for undegenerate configurations with a symmetric $\hat{\mathcal{S}} = \bar{\mathcal{S}} = \bar{\mathcal{S}}^T$.

The extra terms in both Equation (4.25) and Equation (4.26) involve the rotation differential $\delta\bar{\mathcal{R}}$ and introduce significant computational overhead. This is highly inconvenient for interactive simulation, and motivates us to consider the truncation of both $\check{\mathcal{P}}_D = 0$ and $\delta\check{\mathcal{P}}_D = 0$, that can be justified exclusively from the assumption $\delta\bar{\mathcal{R}} = 0$.

It may seem contradictory to define the multi-part displacement expression $\lambda(w, h)$ in Equation (4.7) to guarantee a differentiable $\bar{\mathcal{R}}$ and, afterwards, drop the actual $\delta\bar{\mathcal{R}}$ terms in stress and stress differential expressions. However, even if the $\delta\bar{\mathcal{R}}$ are never used in practice, their existence ensure that $\bar{\mathcal{R}}$ and the derived forces are smooth, not just continuous, which is objectively beneficial. Moreover, in non-interactive applications with weaker efficiency requirements the continuous $\delta\bar{\mathcal{R}}$ may be used to evaluate the exact $\mathcal{P}_D$ and $\delta\mathcal{P}_D$ for fully-implicit integration.

The approximation of stress differentials $\delta\mathcal{P}$ by truncation of rotation differentials has been discussed at length in Section 3.3.4, and does not change the dynamics, only the Newton-Raphson iterates $\Delta\boldsymbol{x}_k^{n+1}$. We can use the modified matrices $\bar{\mathcal{R}}$, $\hat{\mathcal{S}}$ and $\delta\bar{\mathcal{F}}$ in Equation (3.23) to obtain the hybrid approximation $\delta\hat{\mathcal{P}}_D^{(H)}$. The approximation of stress $\mathcal{P}_D \approx \hat{\mathcal{P}}_D$, however, indirectly modifies the elastic forces $\boldsymbol{f}_e \approx \hat{\boldsymbol{f}}_e$ on degenerate elements, and needs to be further analyzed.

4.4.6 Approximate elastic forces

What is the effect of approximating the elastic forces on an finite element? Concerns may be raised in several fronts:

- Realism: is the qualitative realism of the simulations preserved?
- Stability: is the approximation physically stable?
- Robustness: is the approximation numerically stable?

The following aspects are central to the analysis:

Equilibrium configurations: The elastic energy $\Psi_D \geq 0$ has a global minimum when $\epsilon_D = 0$, which corresponds to the subspace of deformations $\mathcal{F} = \bar{\mathcal{R}}\hat{\mathcal{S}}$

with $\hat{\mathcal{S}} = \mathcal{I}$ and arbitrary $\mathcal{R}$. Exact elastic stresses and forces only vanish with $\mathcal{P}_D = 0$ and $\boldsymbol{f}_e = 0$ at such global energy minimum subspace. It can be seen from Equation (4.25) that approximate stresses $\hat{\mathcal{P}}_D$ and forces $\hat{\boldsymbol{f}}_e$ only vanish at exactly the same subspace $\hat{\mathcal{S}} = \mathcal{I}$. As a result, they drive the dynamics to the *exact equilibrium configuration subspace*.

Energy conservation: In order to determine if the approximate elastic forces remain conservative we will try to recover an *hypothetical* approximate elastic energy $\hat{\Psi}_D$ that generates them. If we restrict the discussion to the reference triangle element with $\boldsymbol{x}_1 \in \mathbb{R}^2$, $\boldsymbol{x}_2 = \boldsymbol{r}_2$, $\boldsymbol{x}_3 = \boldsymbol{r}_3$, computing the elastic energy requires the solution of the partial differential equation $\hat{\boldsymbol{f}}_1 = \nabla\hat{\Psi}_D(\boldsymbol{x}_1)$ with the boundary condition $\hat{\Psi}(\boldsymbol{r}_1) = 0$ to ensure that the elastic energy vanishes at the reference configuration $\boldsymbol{x}_1 = \boldsymbol{r}_1$. The numerical solution process is detailed in Appendix A.2, and was performed twice along independent paths. The resulting scalar fields $\hat{\Psi}_D^{X\to Y}$ and $\hat{\Psi}_D^{Y\to X}$ are shown in Figure 4.8. These inferred potential energies are identical in the undegenerate region, but become significantly different in the degenerate region, which contradicts the path-independence of conservative forces. Therefore, the approximate forces cannot be derived from a potential energy, and are not conservative. This result for the reference triangle element trivially extends to the full configuration space $\boldsymbol{x}_e \in \mathbb{R}^{D(D+1)}$ of a D-dimensional simplex element. In consequence, closed trajectories in such configuration space may decrease or, most worryingly, increase, the mechanical energy *iff they overlap the degenerate region*. In absence of damping this last observation would preclude any use of approximate elastic forces. However, in combination with Implicit Euler integration, the spurious energy increases caused by degenerate elements are rapidly absorbed by the intrinsically dissipating time-stepping process. Numerical instability could still manifest for very small integration timesteps, but we have never experienced such problems in our simulations, not even in stress tests with massive degeneration and a tiny $\Delta t = 10^{-4}s$, two orders of magnitude smaller than the value $\Delta t = 1/60s$ commonly used in interactive applications.

Singularities: Approximate stresses $\hat{\mathcal{P}}_D$ and stress differentials $\delta\hat{\mathcal{P}}_D$ do not introduce additional singularities when compared to the exact expressions in Equation (4.25) and Equation (4.26). The continuity analysis in Section 4.4.4 remains valid after the approximation.

Accuracy: Exact (Figure 4.4) and approximate (Figure 4.7) force fields are qualitatively similar. The relative error for the approximate forces on all nodes $\hat{\boldsymbol{f}}_e(\boldsymbol{x}_1)$ is shown in the bottom-right panel of Figure 4.8. The error is small for moderate degenerations, and grows with the magnitude of the non-symmetric part of $\hat{\mathcal{S}}$, which is consistent with the actual approximation $\mathcal{P}_D \approx \hat{\mathcal{P}}_D$ that is ex-

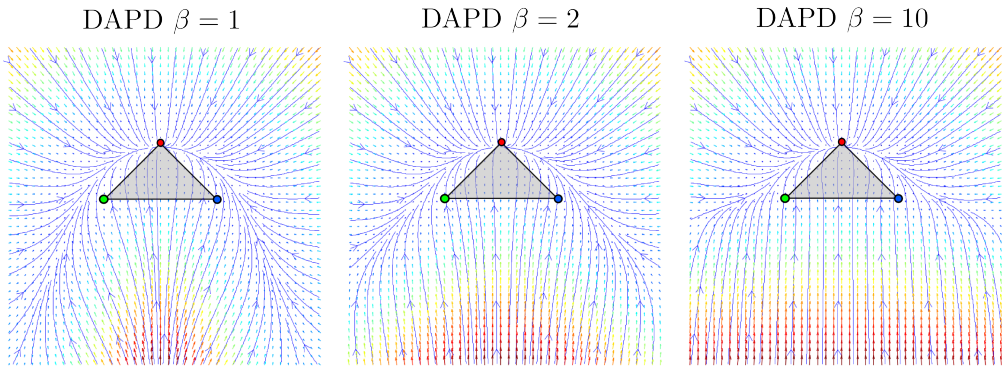

Figure 4.7: Approximate force fields experienced by node $\boldsymbol{x}_1$ (red) in a $[-3, 3]^2$ region of the plane when $\boldsymbol{x}_2$ (green) and $\boldsymbol{x}_3$ (blue) are fixed, for corotational linear FEM with DAPD rotation extraction for different values of β. Degeneracy threshold $\alpha = 0.1$, material parameters $E = 1000Pa$, $\nu = 0.35$.

act for symmetric $\hat{\mathcal{S}}$. Notice the minimal error in a wide centered area directly behind E_{23}, that corresponds to nearly-symmetric scaled reflection deformations $\hat{\mathcal{S}} = \begin{bmatrix} 1 \pm \epsilon & s_{xy} \\ s_{yx} & -s_y \end{bmatrix}$ for $s_{xy}, s_{yx} < \epsilon$. A similar relative error pattern can be seen in the recovered elastic energy fields in Figure 4.8.

Realism is preserved thanks to the unchanged equilibrium configuration and the acceptable accuracy of approximate forces for moderate degenerations. Mechanical energy increase is theoretically possible due to the non-conservative nature of approximate forces, but does not compromise physical stability in practice thanks to the intrinsic dissipation Implicit Euler. No new singularities threaten numerical robustness. We believe that the use of approximate elastic forces is justified for interactive simulation, as its efficiency benefits counterweight its potential drawbacks. In summary:

- Forces are approximate in degenerate configurations, but remain exact otherwise.
- Approximate forces drive the element to the exact equilibrium configuration.
- Approximation error grows with the magnitude of the non-symmetric part of $\hat{\mathcal{S}}$, which is small for moderate degenerations.
- Implicit Euler integration with moderately large timesteps dissipates energy fast enough to avoid stability problems due to non-conservative approximate forces.
- Approximate forces significantly reduce computational cost and are consistent with the common approximate stress differential assumption $\delta\bar{\mathcal{R}} = 0$.

Simulation results for DAPD with approximate forces in persistent and transient degeneration scenarios will be presented in Section 4.5.1 and Section 4.5.2.

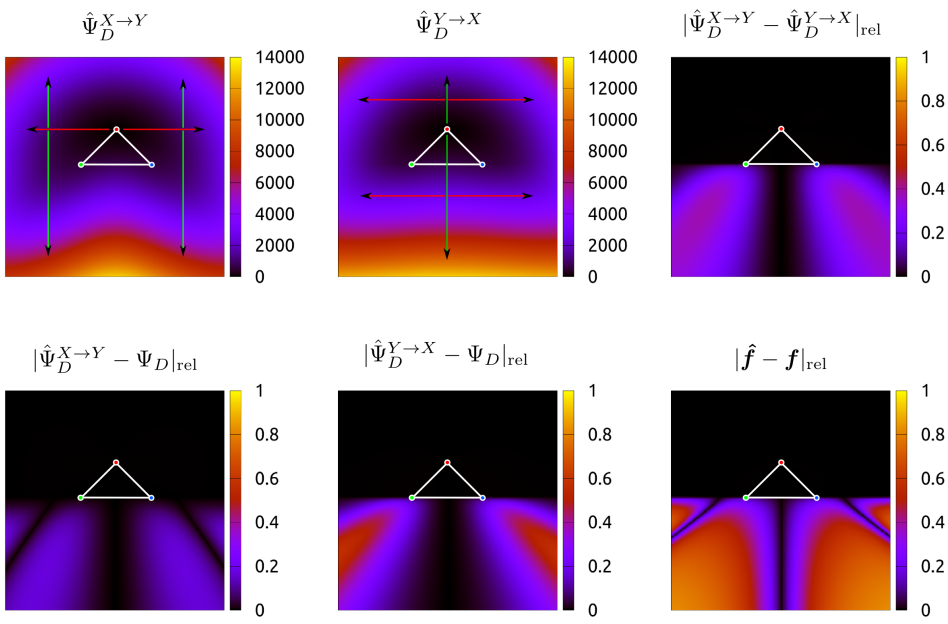

Figure 4.8: Numerical recovery of the approximate elastic energy field induced by approximate forces in a $[-3,3]^2$ region for the example triangle element with material parameters $E = 1000$, $\nu = 0.35$ and DAPD $\alpha = 0.1$, $\beta = 3$. Relative errors are small for moderately degenerate configurations, and grow with the magnitude of the skew-symmetric part of $\hat{\mathcal{S}}$.

4.4.7 A remark on QR factorization

Motivated by our recent analysis of approximate corotational elastic forces in DAPD caused by an asymmetric $\hat{\mathcal{S}}$, we analyzed the forces resulting from corotational FEM with QR factorization $\mathcal{F} = \tilde{\mathcal{R}}\tilde{\mathcal{S}}$, that is also known to generally result in an asymmetric $\tilde{\mathcal{S}}$. We computed the elastic force field using the rotated stiffness matrix $\tilde{\boldsymbol{f}}_e = \tilde{\mathcal{R}}_e\mathcal{K}_e^0(\tilde{\mathcal{R}}_e^T\boldsymbol{x}_e - \boldsymbol{r}_e)$ from Equation (3.17), as often recommended in interactive corotational FEM for efficiency purposes [ESHD05], and tried to recover the energy field along independent paths using the same methodology as in the previous section. The result is shown in Figure 4.9, and reveals that corotational FEM with QR factorization and constant stiffness matices yields non-conservative elastic forces even in undegenerate configurations.

The reason is that the stiffness matrix $\mathcal{K}_e^0$ cannot be considered constant for element deformations decomposed using QR factorization, a fact already noticed in [NPF05], where the authors proposed updating the stiffness matrices every timestep to account for changes in element shapes. Therefore, while the implemention of corotational FEM

using fast and robust QR factorization with precomputed constant $\mathcal{K}_e^0$ may seem computationally efficient, the non-conservative nature of elastic forces must be taken into account. On the other hand, Polar Decomposition and SVD avoid this issue and allow using constant $\mathcal{K}_e^0$ at a higher factorization cost.

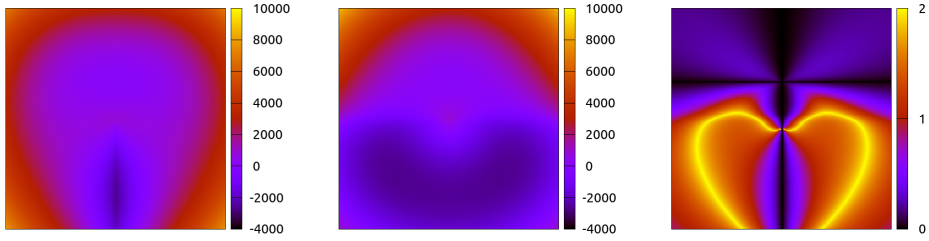

Figure 4.9: Energy fields recovered from a force field induced by QR factorization with a constant stiffness matrix, using $X \rightarrow Y$ (left) and $Y \rightarrow X$ (middle) paths. Notice the large relative error in both degenerate and undegenerate regions (right).

4.5 Results

4.5.1 Results for persistent degeneration

Simulation results for scenes where degeneration is permanently enforced by gravity are shown in Figure 4.10 and Figure 4.11. While DAPD with $\beta = 1$ is equivalent to the Project method in [CFS14], larger $\beta = 5$ clearly improves persistent degeneration resistence thanks to stronger and coherently aligned elastic forces. Increasing β also reduces force smoothness across the transition region between regular and inverted configurations $0 \leq w < h$ in Equation (4.7). In our experiments, we obtained the best tradeoff between degeneration resistance and smoothness for $\beta \in [3, 5]$.

4.5.2 Results for transient degeneration

Some benchmarks of the temporal evolution of transient degeneration can be seen in Figure 4.12. For massive, random degeneration tests, in the top row of the figure, DAPD recovers much faster than QR and ISVD. In case of violent collision tests against a flat ground, however, the benefit of DAPD over ISVD is less noticeable, as elements tend to collapse, but do not significantly invert, and therefore ISVD and DAPD produce very similar results. Both are superior to QR, that suffers from large self-induced inversions due to the critical point located exactly at a collapse configuration (see third panel on the top row of Figure 4.2, as detailed in [CFS14]. Both random degeneration and collision benchmarks show little difference between DAPD $\beta = 1$ and $\beta = 5$. This can be explained by the fact that fast, transient degeneration does not benefit from

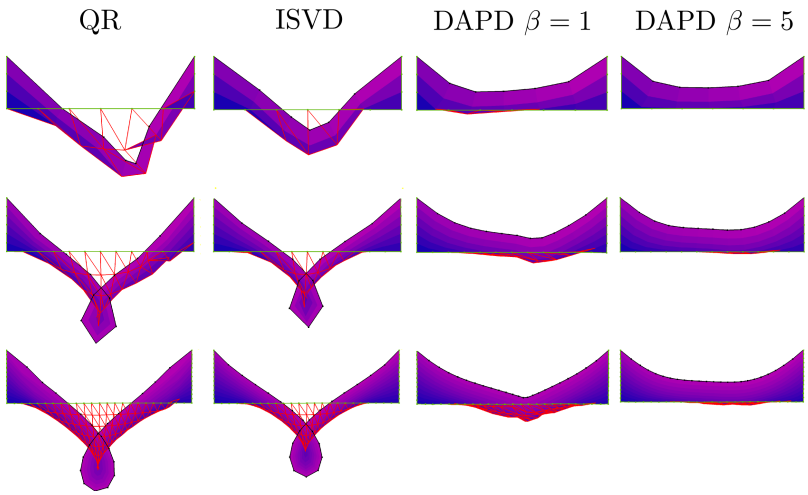

Figure 4.10: Equilibrium configuration resulting from the dynamic simulation of a 2D Beam supported by the left, right and bottom sides, using different discretizations (rows) and degeneration treatment methods (columns). Discretizations with 144, 576 and 2000 triangle elements. Degenerate elements shown in red wireframe. Solver parameters: $\Delta t = 1/60s$, $n_{\text{NR}} = 1$, $n_{\text{LS}} = 10$. Material parameters: $E = 1000Pa$, $\nu = 0.25$, $m = 5kg$. Gravity $g = -100m/s^2$. DAPD parameters: $\alpha = 0.1$.

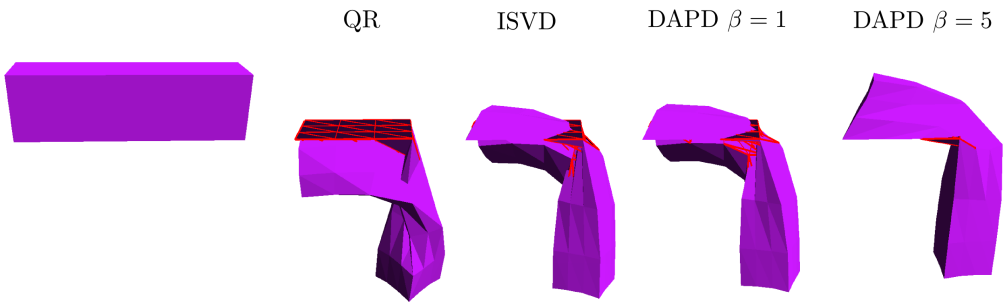

Figure 4.11: Equilibrium configuration resulting from the dynamic simulation of a 3D Beam supported by the left half of its base using different degeneration treatment methods (columns). Discretization with 324 tetrahedron elements. Degenerate elements shown in red wireframe. Solver parameters: $\Delta t = 1/60s$, $n_{\text{NR}} = 1$, $n_{\text{LS}} = 10$. Material parameters: $E = 1000Pa$, $\nu = 0.25$, $m = 5kg$. Gravity $g = -50m/s^2$. DAPD parameters: $\alpha = 0.1$.

stronger degeneration resistance as much as static equilibrium configurations do, for exemple, in Figure 4.10 and Figure 4.11.

	mass	E	ν	Elements	Nodes
Cube666	$1kg$	$1000Pa$	0.35	1296	343
Dragon1K	$1.3kg$	$1000Pa$	0.35	411	163
Armadillo1K	$1.2kg$	$10000Pa$	0.35	452	201
Horse1K	$7.5kg$	$100000Pa$	0.35	283	142

Table 4.1: Models used in degeneration treatment benchmarks in Figure 4.12.

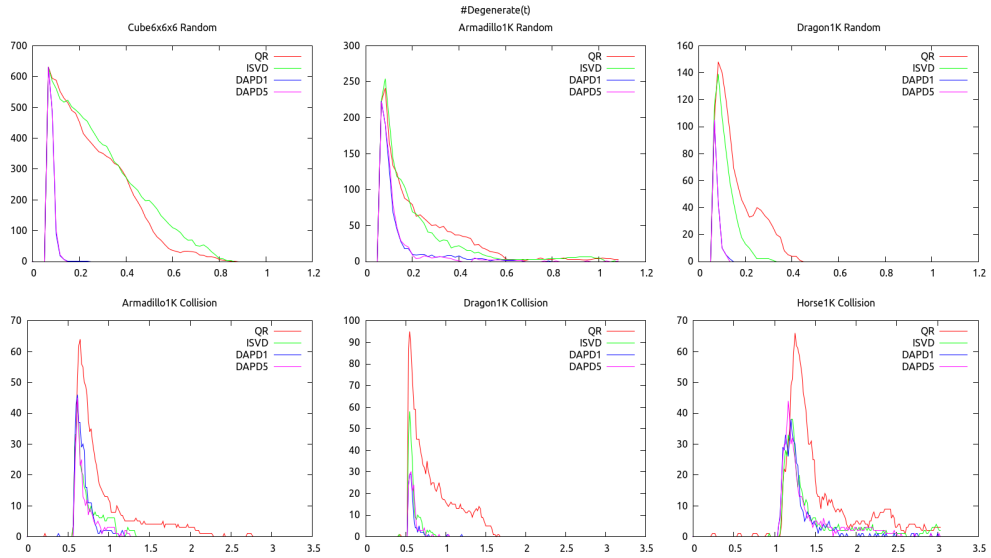

Figure 4.12: Evolution of the number of degenerate elements using different degeneration treatment methods. Top row shows recovery from a random perturbation of the nodes inside the model bounding box for the Cube666, Armadillo1K and Dragon1K models (Table 4.1. Bottom row: elastic objects Armadillo1K, Dragon1K and Horse1K fall and collide against flat ground from a height of $5m$ pulled by gravity $g = -100m/s^2$, with many elements degenerating as a result. DAPD1 and DAPD5 use $\beta = 1$ and $\beta = 5$ respectively.

4.6 Discussion

We presented *degeneration-aware polar decomposition*, a method that generalizes and improves the results in [CFS14]. The generalized method guarantees the smothness of $\bar{\mathcal{R}}$, which benefits convergence in quasi-implicit integration, and allows its application to fully-implicit integration.

Reinterpreting the modified $\bar{\mathcal{R}}$ for degenerate elements as a modified strain tensor ϵ_D provided deeper insight and revealed that approximate elastic forces in degenerate elements are not strictly conservative but do not cause stability problems in implicit integration, a discussion omitted in [CFS14].

CHAPTER 5

Contact Determination

In this chapter we present a fast contact determination scheme for intersecting deformable solids with detailed surface geometry. Given a high resolution closed surface mesh we automatically build a coarse embedding tetrahedralization and a partitioned representation of the surface in a preprocess. During simulation, the contact determination algorithm finds all intersecting pairs of deformed triangles using a memory-efficient barycentric bounding volume hierarchy, connects them into potentially disjoint intersection curves and performs a topological flood process on the exact intersection surfaces to discover a minimal set of contact points. A unique contact normal is computed for each contact volume, based on a continuous definition of surface normals, and used to find contact point correspondences suitable for contact treatment.

5.1 Introduction

Physically based simulation of rigid and deformable objects is common in computer graphics, where the focus is on robustness, controllability, visual spectacularity and computational efficiency over physical accuracy. Robustness and efficiency are critical for interactive simulation, and drive commercial videogames to use the simplest physical models possible while rendering the most detailed appearance models possible. This strategy often extends to collision detection, where primitive or extremely simplified geometry is used as a coarse approximation of the visualized shapes, causing visible gaps in contact configurations.

In such context, efficient contact determination for deformable geometry poses spe-

cific challenges, as standard acceleration data structures need to be dynamically updated to reflect the deformed configuration, and contact determination cost increases with geometry complexity. Efficient algorithms that detect surface proximity exist, but perform poorly if objects intersect, a situation that cannot be strictly avoided in interactive simulation. Intersection can be alleviated at a higher CPU cost using continuous collision detection (CCD), but strict update rate requirements limit its applicability in interactive simulations. Contact determination between intersecting surfaces at discrete time steps is more involved than proximity detection, but necessary to ensure robustness. In addition, high resolution surfaces often exhibit high frequency detail and irregular normals that further complicate the fast computation of meaningful contact information.

We propose an efficient contact determination scheme that enables simulation of intersecting, contact-interacting deformable solids with highly detailed geometry at interactive rates. The main contributions of our approach are:

- A CD-specific surface representation that accurately matches visual geometry detail and accelerates contact determination.
- A memory-efficient BVH structure defined in barycentric coordinates that reduces update cost under deformation.
- A flood-based contact point generation algorithm that computes a minimal set of contact points efficiently.
- A novel definition of contact normals that is robust in presence of high frequency surface detail and severe interpenetrations.

The resulting method can be used to handle all contact determination between deformable objects, if allowed to interpenetrate slighly, or combined with an exterior proximity-based approach as a robust fallback strategy for unavoidable interpenetrations.

5.2 Related work

Contact determination and treatment between deformable solids is essential in physics-based animation and has been subject to extensive research. A good yet ageing survey on the field is available [TKH+05]. Virtually all existing approaches use either dynamic bounding volume hierarchies or spatial classification structures to localize close or overlapping geometry. Several hierachical schemes that avoid full recomputation on deformation have been proposed, most notably: BD-Trees [JP04], and BN-Trees [SGO09], that respectively introduce efficient refit strategies for bounding volumes and surface

normals, by computing approximate tight bounds on high resolution geometry under a combination of linear deformations.

Alternative proximity-based methods have been proposed based on deformed distance fields [FL01], using the GJK algorithm on non-rigid vertices [GFT08], and adaptive stochastic sampling of surface geometry [KNF04].

Severe interpenetration has been specifically addressed for cloth and solid simulation. A method for untangling cloth with global intersection analysis was presented in [BWK03]. For solids, consistent depth estimation in [HTK+04] addresses the issues of proximity-based methods in case of deep interpenetration, and computes consistent normals and depth by volumetric propagation of robust values on the surface. In [HFR08] contacts are found by ray-tracing inwards from the surface of the objects along the surface normals, a method that only works reliably for convex intersection volumes. Collision detection for highly detailed surfaces embedded in tetrahedral meshes with changing topology was shown in [WT08b], using simulation tetrahedrons as coarse bounding volumes for the embedded surface vertices. Fully volumetric contact determination at arbitrary resolution using GPU rasterization has been demonstrated in [AFC+10]. A CPU-based volumetric CCD approach for high resolution tetrahedral meshes was introduced in [TMY+11].

5.3 Deformable solid geometry

In order to simulate contact-interacting deformable solids with the highest possible surface detail at interactive rates we choose to completely decouple the simulation geometry *SIM*, the collision detection geometry *CDG* and the visualization geometry *VIZ* (Figure 5.1). Simulation geometry will be represented as a volumetric tetrahedralization. The visualization and collision detection geometries will be barycentrically embedded in the non-conforming and strictly bounding tetrahedralization.

This will enable us to use the coarsest possible volume discretization for simulation and the finest possible geometry for visualization and collision detection. Achieving the same level of visualization and collision detail with a conforming tetrahedralization would be unfeasible, as for an object size S and discretization length h, volume complexity $\mathcal{O}((S/h)^3)$ grows much faster than surface complexity $\mathcal{O}((S/h)^2)$. Adaptive conforming tetrahedralization can alleviate this issue generating larger tetrahedrons far from the surface [ACSYD05, LS07], but produces small tetrahedrons near the surface that can easily degenerate on contact [CFS14] and reduce the convergence of iterative solvers [MZS+11b].

Decoupling *CDG* from *SIM* allows highly detailed collisions, avoiding unrealistic visual gaps between the surfaces of contact-interacting solids while limiting the simulation cost. Decoupling *CDG* from *VIZ* allows us to represent the same surface in

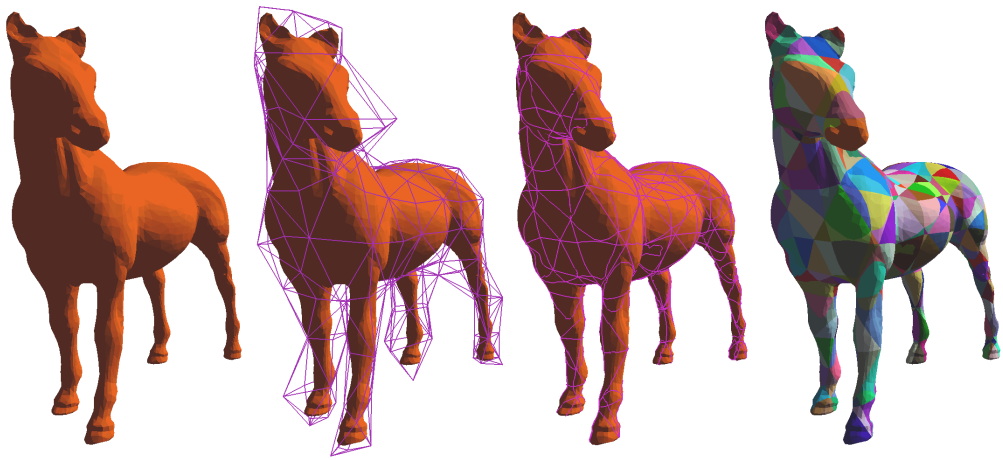

Figure 5.1: A source model with 10K triangles (left), its automatic coarse tetrahedralization (mid-left), the clipped mesh (mid-right) and the final CDG representation with random patch colors.

a specifically optimized format that discards any appearance parameters but includes mesh topology, unnecessary for most rendering schemes, as well as additional precomputed data detailed in section 5.4 that will help reducing contact determination cost.

5.3.1 Tetrahedralization

We build a non-conforming, strictly bounding, coarse tetrahedralization of the visualization mesh using an approach similar to [SLJB13] with a user-defined tetrahedron feature size h. The high-level tetrahedralization process performs the following tasks

1. Construct a regular tetrahedralization of the AABB of the visualization mesh. The AABB is partitioned into cubic cells of side length h, and each cell is minimally split into 5 tetrahedrons, taking parity into account to ensure topology compatibility between neighbour cells [CMS01].

2. Next, tetrahedrons are classified into External, Internal, and Piercing. Piercing tetrahedrons intersect the visualization mesh. Completely external tetrahedrons are deleted, while Internal and Piercing remain.

3. Then, piercing tetrahedrons are processed to detect features (faces and edges) that are shared between different tetrahedrons but do not pierce the surface themselves. These features are split to avoid artificial tetrahedron neighbourhood due to discretization length.

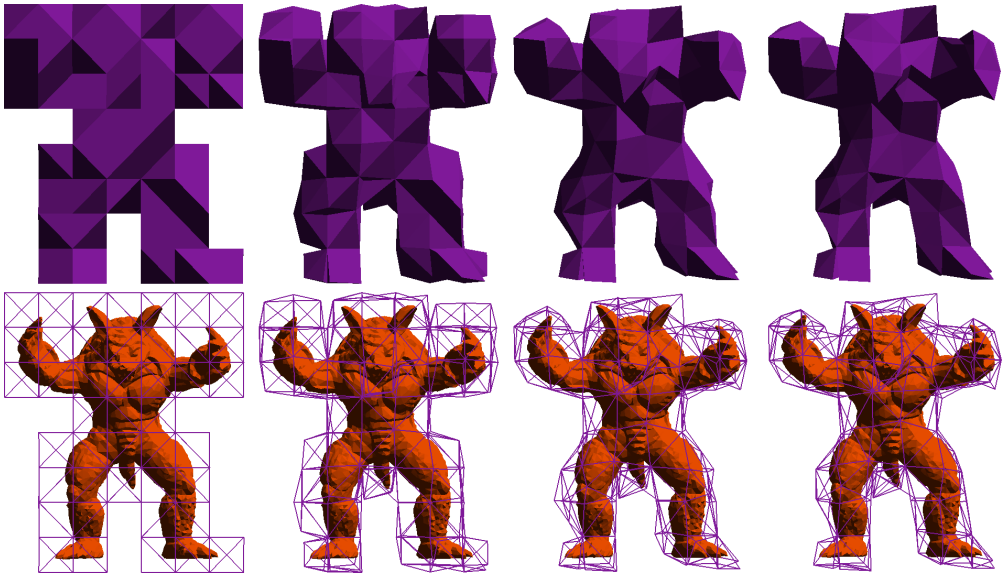

Figure 5.2: Coarse tetrahedralization of Armadillo10K model with 0, 1, 5, and 10 fitting steps, from left to right, and a relaxation coefficient $k_{\mathrm{ODT}} = 0.1$. The arms are fitted independently despite being initially connected to the head in the regular tetrahedralization, thanks to feature splitting.

4. Finally, external nodes of Piercing tetrahedrons are iteratively adjusted to fit the visualization surface while remaining strictly external.

The last step is the most involved. Each global fitting pass iterates over all external nodes $\boldsymbol{x}_i$ of piercing tetrahedrons and computes the displecement that minimizes the local Optimal Delaunay Triangulation (ODT) energy [ACSYD05]. This displacement is relaxed with a user-defined coefficient $k_{\mathrm{ODT}} \in (0, 1]$. Each node displacement affects its adjacent 1-ring of external tetrahedron faces, that need to be tested for collision against the visualization surface. The displacement is stopped *before* the first intersection to guarantee a strictly bounding tetrahedralization. Once all nodes have been sequentially displaced the whole process is repeated up to a user-defined number of fitting iterations.

This mesh generation method prioritizes well-shaped tetrahedrons over accurate reproduction of embedded surface geometry, as it will be used for simulation but never directly for collision detection. While we do not enforce strict topology compatibility with the visualization mesh in order to minimize the number of tetrahedrons, the feature split in step 3 greatly improves fitting in step 4 (Figure 5.2). The whole-process runs fast and can be performed automatically as a preprocess. In a videogame asset production pipeline the automatically generated meshes could be manually adjusted afterwards to match application specific criteria and restrictions.

5.3.2 Barycentric embedding

Visualization and collision geometry are barycentrically embedded in simulation tetrahedrons, using the same interpolation scheme as the material discretization in Section 2.2.1.1. Barycentric deformation is affine and therefore preserves linear features. It can be efficiently applied to the vertices that define segments and triangles in the reference configuration $\boldsymbol{v}_i^m$ to obtain their deformed positions $\boldsymbol{v}_i^s$

$$\mathcal{B} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ \boldsymbol{x}_1 & \boldsymbol{x}_2 & \boldsymbol{x}_3 & \boldsymbol{x}_4 \end{bmatrix}, \quad \begin{bmatrix} 1 \\ \boldsymbol{v}^s \end{bmatrix} = \mathcal{G}_m^s \begin{bmatrix} 1 \\ \boldsymbol{v}^m \end{bmatrix}$$

Where $\mathcal{G}_m^s = \mathcal{B}_s \mathcal{B}_m^{-1}$ is the barycentric deformation defined by the tetrahedron nodes $\boldsymbol{x}_j$. Visualization geometry can be transformed directly in the GPU using a convex combination of weighted tetrahedron nodes $\boldsymbol{v}_i^s = \sum_j^4 \xi_j \boldsymbol{x}_j$, where $\boldsymbol{\xi}$ are the constant barycentric coordinates. This results in a constant number of floating point operations to deform a vertex, which is highly convenient for efficiency. Collision geometry vertices only need to be transformed when required by the contact determination algorithm.

Barycentric embedding has numerous advantages: deformation $\boldsymbol{x} = \phi_e(\boldsymbol{X})$ is trivially inverted $\boldsymbol{X} = \phi_e^{-1}(\boldsymbol{x})$ using $(\mathcal{G}_m^s)^{-1}$, and the barycentric weights of an arbitrary point $\boldsymbol{p}$ can be computed as $\boldsymbol{\xi} = \mathcal{B}_s^{-1}\boldsymbol{p}$. This will be used to efficiently express arbitrary contact points in barycentric coordinates, required for contact response. In addition, affinity ensures that linear features are preserved, and that contact determination will only need to perform basic geometric tests on triangles, avoiding curved features.

On the negative side, barycentric embedding is limited to C^0 continuity across simplex boundaries, which can result in a faceted appearance in some situations, as seen in Figure 5.3. Higher continuity alternatives exist: Moving Least Squares (MLS) embedding [KMBG09] and higher order simplex elements (eg. quadratic tetrahedrons [BC14]). Both alternatives are significantly more expensive. In case of MLS, deformation of embedded points involves the linear combination of a variable number of node positions $N \geq 4$, and $\phi_e^{-1}(\boldsymbol{p})$ requires solving a system of non-linear equations. In case of higher order tetrahedrons, embedded point deformation involves a non-linear combination of node positions, and $\phi_e^{-1}(\boldsymbol{p})$ requires solving a non-linear system of equations. In addition, neither alternative preserves linear features and geometric tests with curved triangles for contact determination. Moreover, neither MLS nor higher order tetrahedrons remain bounded by the coarse tetrahedralization (Figure 5.3). Higher continuity methods are attractive for offline or dedicated high-quality simulation [Rot02], but barycentric embedding remains the most efficient option for CPU-limited interactive simulation.

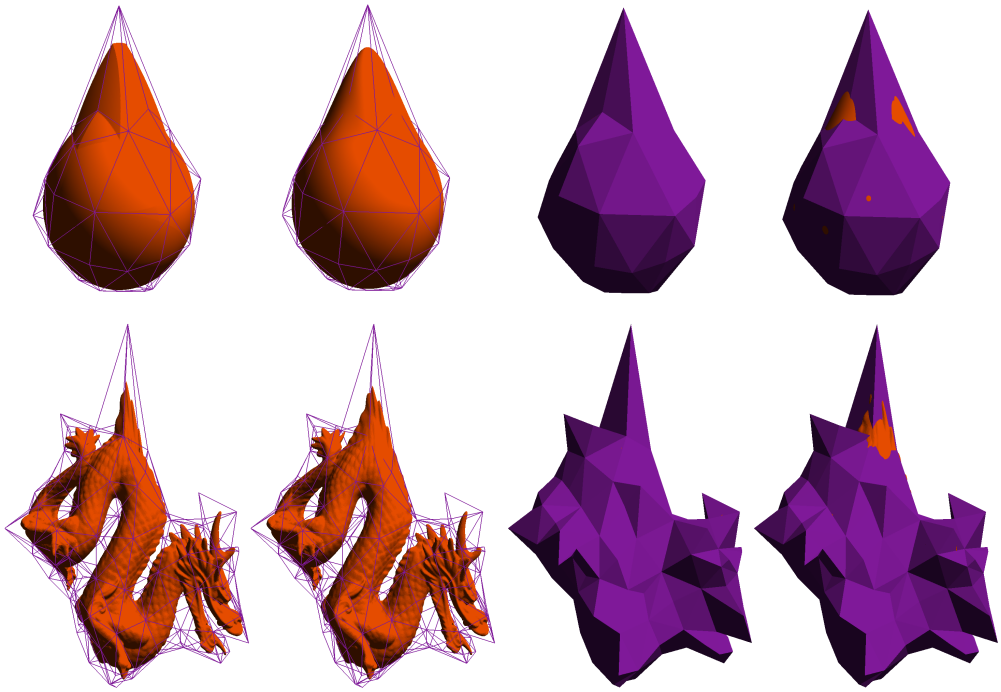

Figure 5.3: Comparison between Barycentric (left) and MLS (right) embedded meshes, in orange. For smooth surfaces (sphere with 327680 triangles) embedded in coarse tetrahedral meshes, barycentric embedding produces visible C^0 artifacts. For arbitrary surfaces (Dragon with 100K triangles) the artifacts are less noticeable. MLS embedding is not strictly contained by the bounding tetrahedralization, in violet, while barycentric embedding is.

5.4 Contact determination geometry representation

We propose a collision geometry representation that results from partitioning the detailed visualization geometry into surface patches interior to single tetrahedrons of *SIM* (Figure 5.1). This will allow efficient and completely decoupled processing of each tetrahedron and its embedded surface patches, vertices and triangles during collision detection. For detailed surface geometry with several orders of magnitude more features than its simulation geometry this will prove an efficient choice, as it enables precomputation of per-patch BVH and other acceleration structures, guarantees good data locality and simplifies several low-level optimizations.

5.4.1 Surface partitioning

Given a detailed surface mesh *VIZ* and its embedding coarse tetrahedralization *SIM*, the *CDG* is built in a preprocess as follows: We clip each *VIZ* surface triangle against

SIM tetrahedron faces and triangulate the resulting convex polygon if necessary, reconstructing its neighbour topology. The result is a valid triangle mesh where no triangle spans across different tetrahedrons. The clipped mesh is then partitioned into a set of simply-connected open triangle patches $\{P_i\}$ that lie completely inside a single tetrahedron, with boundary edges coplanar to its faces. Triangle and patch neighbour topology is computed and stored to perform surface navigation at both levels during contact determination. Addidionally, we also precompute the centroid, vector area and scalar area of each patch, required at runtime as detailed in section 5.5.3.2. Finally, we compute a static bounding volume hierarchy for the triangles that form the patch as detailed in section 5.4.3. Once all patches in a tetrahedron have been processed, we compute and store the bounding volume of all the vertices in all its patches, as detailed in section 5.4.2.

The resulting *CDG* contains an entry for each tetrahedron in *SIM* that contains at least 1 patch with at least 1 triangle. A *CDG* tetrahedron E can contain any number of patches formed by any number of vertices and triangles. The *CDG* is stored in an indexed structure-of-arrays that contains plain-old-data descriptors for Tetrahedrons, Patches, Triangles and Vertices, which enables trivial serialization and fast runtime access. Triangle and vertex data is arranged in sequential sub-arrays per-tetrahedron, and per-patch within it, ensuring good locality for the memory acces patterns of the contact determination algorithm. We duplicate vertices across patch boundaries to achieve the desired sequential layout and allow efficient barycentric transformation of all vertices in a patch using SIMD with optimal cache behaviour.

We initially considered and discarded the alternative approach of classifying, as opposed to partitioning, surface features into tetrahedrons [WT08b]. Vertices could be safely classified into single tetrahedrons, but edges and triangles might span any number of tetrahedrons. This would force us to consider an arbitrary number of tetrahedrons and all their classified features during contact determination. Additionaly, non-partitioned linear features with vertices classified into different tetrahedrons could exit the whole tetrahedralized volume under deformation, which would invalidate the efficient BVH strategy described in section 5.4.3.

5.4.2 Barycentric Discrete Orientation Polytope (b-DOP)

A Barycentric Discrete Orientation Polytope (b-DOP) is a bounding volume analogous to an axis-aligned bounding box (AABB) defined in barycentric coordinates, that can also be seen as a D-dimensional k-DOP [KHM+98] with $k = 2(D + 1)$, defined by $(D + 1)$ slabs along directions perpendicular to simplex faces in cartesian coordinates (Figure 5.4). Given the simplex vertices, the b-DOP can be compactly stored as $D+1$ barycentric intervals $[a_i, b_i] \in [0, 1]$, requiring $k = 2(D + 1)$ floating point values.

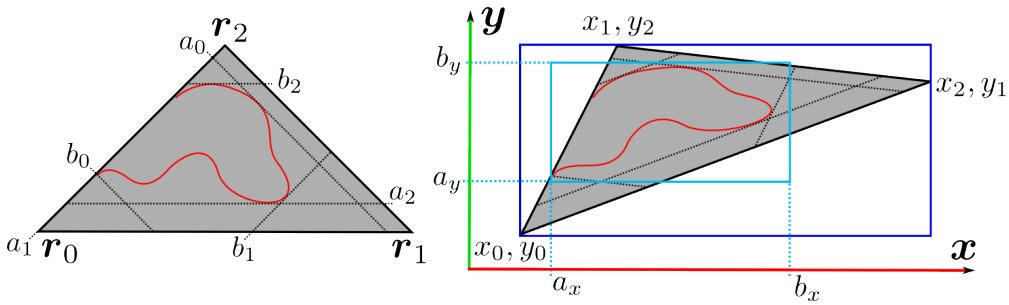

Figure 5.4: Left: b-DOP in reference configuration. Right: Deformed simplex AABB (dark blue), and tight AABB (light blue) computed from sorted coordinates $x_0 < x_1 < x_2$ and $y_0 < y_1 < y_2$.

The b-DOP of barycentrically embedded geometry remains constant under deformation, thus, for each tetrahedron we can precompute the b-DOP of its embedded vertices, store it in the *CDG* and reuse it during simulation without refitting to efficiently discard raycast and intersection queries on all patches inside a deformed tetrahedron. For raycasts, we transform the ray with the inverse barycentric matrix $\mathcal{B}_s^{-1}$ of the tetrahedron and perform ray-vs-slabs tests in barycentric coords. The required $\mathcal{B}_s^{-1}$ are cached and reused in several parts of the contact determination algorithm.

Reusing the b-DOP for overlap tests between a pair of deformed tetrahedrons E_1, E_2 is more involved, as their axis are neither aligned in each other's barycentric coordinates nor in global coordinates. An exact separating axis test in global coordinates is possible, but extremely expensive due to the large number of face and edge-pair axis from b-DOP polyhedrons, that would also need to be computed explicitly. Instead, we use the fast tight bounds approximation of [OGRG07] to find a global bounding volume (AABB or 14-DOP) for each deformed tetrahedron b-DOP (Figure 5.4), and test them for overlap. We reproduce the relevant formula from [OGRG07] for further reference. Given a direction x and the sorted element vertex positions $x_i \leq x_{i+1}$, the conservative global bounds $[a_x, b_x]$ of the b-DOP along x are:

$$a_x = b_0 x_0 + a_3 x_3 + a_2 x_2 + (1 - b_0 - a_3 - a_2) x_1, \tag{5.1}$$

$$b_x = b_3 x_3 + a_0 x_0 + a_1 x_1 + (1 - b_3 - a_0 - a_1) x_2. \tag{5.2}$$

5.4.3 Surface-patch b-DOP-Trees

In order to accelerate raycast and intersection queries with all triangles inside a patch we precompute a binary b-DOP-Tree structure with implicit topology that can be stored in with zero memory overhead and has several useful properties. To design such structure, we observe that the triangle sub-array associated with a single patch can be ordered arbitrarily, and exploit this fact to arrange triangles in consecutive sub-arrays

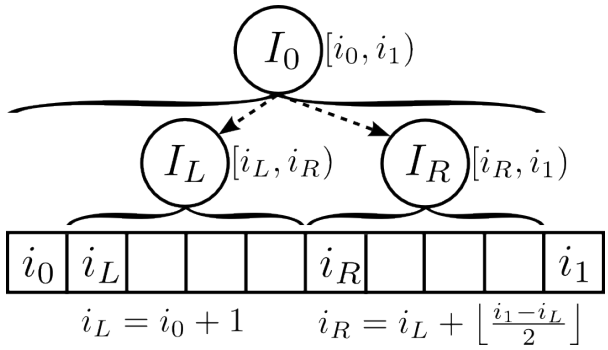

Figure 5.5: *b*-DOP-Tree sub-range layout

per BVH node, so that they only need to store the index sub-ranges $[i_0, i_1)$ of their triangle sub-arrays. For a binary tree the index sub-ranges can be made implicit by using a median-split top-down construction strategy that uniquely determines the left $[i_0, i_0 + \left\lfloor \frac{i_1 - i_0}{2} \right\rfloor)$ and right sub-ranges $[i_0 + \left\lfloor \frac{i_1 - i_0}{2} \right\rfloor, i_1)$ of a given node range $[i_0, i_1)$.

At this point BVH nodes are completely implicit except for their associated bounding volume geometry, a *b*-DOP in our case, that needs to be stored. By definition, *b*-DOP intervals must lie inside $[0, 1]$, a very small interval that does not need the full precision of floating point numbers, thus, we quantize *b*-DOP min/max values with 8 bits so that a whole *b*-DOP uses only 64 bits to store its 4 intervals. The triangles in a *CDG* store 3 vertex indices and 3 neighbour triangle indices, that need to be global to allow inter-patch surface navigation. If we use 32 bit unsigned integers to store these indices, and accept limiting the total number of vertices and triangles to 2^{21}, we free $6 \times 11 = 66$ bits that can be used to store a quantized *b*-DOP per triangle.

Combined with the implicit hierarchy definition, this allows embedding all BVH node information in a single triangle. Triangle descriptors are simultaneously BVH node descriptors, and we arrange them so that the first triangle in each sub-range $[i_0, i_1)$ represents both the triangle itself, at i_0, and the BVH node that bounds i_0 and its left $[i_L, i_R)$ and right $[i_R, i_1)$ sub-trees, with $i_L = i_0 + 1$ and $i_R = i_L + \left\lfloor \frac{i_1 - i_L}{2} \right\rfloor$. At each level of the recursive top-down BVH build process we sort triangles according to the median-split criteria and the first triangle is chosen as the BVH node. This triangle must not be included in the recursive treatment of its sub-trees ranges, as it is no longer free to be reordered due to the implicit hierarchy definition. We are free to choose any median-split criteria and classify triangles accordingly. We select the longest *b*-DOP axis, but alternative heuristics such as SAH could be used.

The result is essentially a complete hierarchy where triangles are nodes with implicit topology that store quantized *b*-DOP in a bit-packed format that requires zero extra memory usage for *CDG* meshes with up to $2^{21} \approx 2M$ features and adds a very

small CPU unpacking overhead during hierarchy traversal. The b-DOP-Tree remains constant in barycentric coordinates and does not need to be refit when its embedding tetrahedron deforms. Its memory layout (Figure 5.5) allows any triangle sub-array to be considered a "leaf" BVH node, which can be used during raycast and intersection queries to stop recursion at a given threshold $i_1 - i_0 \leq T_{\text{leaf}}$ and process $[i_0, i_1)$ sequentially using hand-optimized code (eg: SIMD). This threshold can be set dynamically during recursion or fit to an optimal value for a specific hardware architecture.

A related approach was used in [EBM12] that requires a minimum of $N = 2$ maximally separated triangles at each tree node to implicitly span the whole subtree AABB. During recursive queries the AABB need to be evaluated with $3N$ random accesses to the vertices of the N triangles. Moreover, the N maximally separated triangles that span the AABB of a BVH node must be individually tested when such node is visited during a raycast or intersection query, as they do not appear deeper in the tree, which can be innefficient as such triangles are not furtherly culled by the hierarchy (eg: the N triangles that span the root node interval are by definition placed at opposed ends of the global AABB but nevertheless need to be individually tested).

5.4.3.1 Raycast on a b-DOP-Tree

We transform the ray into barycentric coordinates in the parent tetrahedron and perform all recursive b-DOP tests in that coordinate space, converting the results to cartesian coordinates when a triangle hit is found. In our non-SIMD ray-vs-triangle implementation we obtained best results with a fixed leaf threshold $T_{\text{ray}} = 1$.

5.4.3.2 Intersection of two b-DOP-Trees

In order to find all intersecting triangle pairs on two patches P_i, P_j embedded in two tetrahedrons E_i, E_j we perform a standard dual-hierarchy recursive test on the patch b-DOP-Trees with a descend-largest strategy to guide the recursion. At each level a pair of b-DOP needs to be tested for intersection. In addition to the tight AABB, we use Eqs. 5.1,5.2 to find tight slabs along the vector area directions of P_i, P_j (defined in Section 5.5.3.1, Eq. 5.3), that greatly improve the culling at higher levels of the hierarchy for low curvature surface geometry. Tetrahedron vertex projections need to be sorted once per axis, but remain constant during recursion.

In a standard dual-hierarchy test, recursion stops when leaf nodes are reached on both sides with T_i and T_j triangles that require $T_i \times T_j$ exact intersection tests. Instead, we exploit the fact that any sub-range can be used as a leaf to stop recursion at any level of the hierarchies when $T_i \times T_j \leq T_{\text{int}}$, and obtained best results for $T_{\text{int}} = 8$. Further optimizations such as front caching [SGO09] are possible and left as future work.

5.4.4 Nested BVH

A two-level nested BVH strategy is used for raycast and intersection tests on deformable solids. A coarse global BVH for *SIM* tetrahedrons is tested in a first phase, and, for any overlapped tetrahedrons found, the disjoint b-DOP-Trees of all its internal patches are tested in a second phase. This two-level BVH scheme is also adequate for non-deformable solid objects with detailed surface geometry. We experimented with several standard bounding volumes (Spheres, AABB and 14-DOP), and found that a complete hierarchy of 14-DOP results in the fastest overall results. For a coarse tetrahedron mesh with less than a thousand tetrahedrons the increased refit cost of k-DOP is outweighted by their superior culling performance. Leaf tetrahedron 14-DOP only need to bound its internal patches and can be efficiently refit using the precomputed tetrahedron b-DOP and the fast tight bounds in Eqs. 5.1,5.2. A more sophisticate refit strategy could be used for finer meshes.

The coarse BVH topology does not change at runtime and can be precomputed and stored in the *CDG*. The dynamically refit 14-DOP are stored within the deformable solid and accessed through the static hierarchy definition. For simple *SIM* with up to 1000 tetrahedra a brute-force $\mathcal{O}(n^3)$ bottom-up build strategy that minimizes BVH node volume is feasible and yields notably tighter bounds under global solid deformation.

5.5 Contact determination algorithm

In physically-based simulation, contact determination is the process of computing the geometric information required to avoid or correct the interpenetration of solid objects. This geometric information is generally used to define inequality contact constraints $g \geq 0$. We do not assume any specific constraint satisfaction method and focus on the generation of geometric information suitable for any approach (eg: penalty forces [TMOT12], impulses [BFA02], position-based dynamics [MHHR07], LCP [DAK04]).

Contact constraints on a pair of solids Ω_A, Ω_B in non-penetrating configuration can be defined by means of a scalar gap function $g = (\boldsymbol{p}_A - \boldsymbol{p}_B)^T \hat{\boldsymbol{n}}_B \geq 0$, where $\boldsymbol{p}_A, \boldsymbol{p}_B$ are the locally closest points on the object surfaces Γ_A, Γ_B, and $\hat{\boldsymbol{n}}_B$ is the normal direction on the *master* surface Γ_B, parallel to the gradient of the distance function $d(\Gamma_A, \Gamma_B)$ at $\boldsymbol{p}_B$ [Wri06]. The gap function definition can be extended to return the penetration depth $g < 0$ when solids intersect. For some contact treatment approaches the surface area S associated with each contact point is also useful to reduce the dependency on surface discretization.

The standard approach for contact determination between non-penetrating solids is finding all pairs of geometric features that have closest points within a distance threshold d_ϵ. For surface geometry represented with triangle meshes the closest points

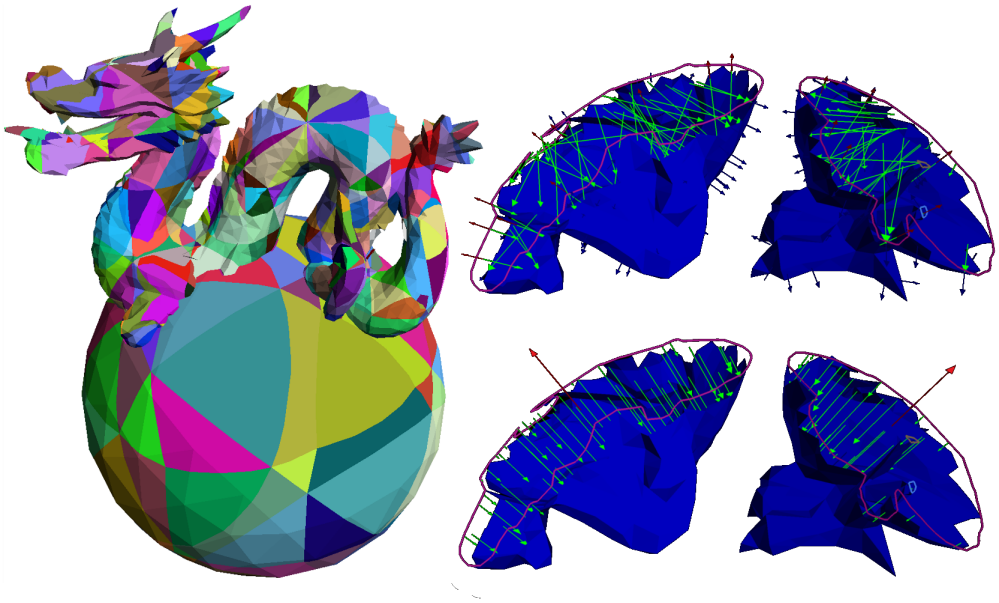

Figure 5.6: Left: A dragon with 5000 triangles and a ball with 1500 intersect in 2 disjoint volumes. Right Top: Local surface normals yield globaly incoherent directions. Right Bottom: Vector area contact normals defined by each intersection curve produce globally coherent contact point correspondences that are better suited for smooth contact response.

are realized by vertex-face (VF) and edge-edge (EE) feature pairs. The proximity-based approach produces generally smooth contact normals and can be implemented efficiently.

Unfortunately, in many applications, specially in interactive simulation, strict non-penetration cannot be guaranteed. The proximity-based computation of $\boldsymbol{p}_A, \boldsymbol{p}_B, \hat{\boldsymbol{n}}_B$ breaks down for intersecting solids, as globally near points on Γ_A, Γ_B may be far, or topologically disconnected, if they lie on the boundary of one of the potentially many disjoint intersection volumes, resulting in invalid contact constraints that fail to correct the interpenetration. The consistent depth estimation approach introduced in [HTK+04] reduces these issues by propagating surface normals into the intersection region and defining a smooth penetration depth field on the object interior, sampled at the simulation nodes. The algorithm is best suited for coarse surface geometry, due to its dependency on surface normals evaluated at specific edge/triangle intersection points. Such local normals may not robustly approximate the surface for detailed geometry with high frequency features, as can be seen in Figure 5.6.

Contact constraints can only act directly on the simulation degrees of freedom, in

our case the tetrahedral mesh node positions. For a general contact configuration we cannot assume that contact points will be coincident with such node positions, and the effect of a contact constraint on the simulation DOF needs to be considered. In our case, the contact points on the barycentrically embedded *CDG* surface need to be specified in barycentric coordinates in their parent tetrahedron, as detailed in Section 3.4.2. For detailed surfaces, a standard contact determination method will produce a large number of contact points (eg: one for each interior vertex) that ultimately act on a few simulation DOF. This increases the computational cost and, for constraint-based contact treatment, results in a severely overconstrained system that may only be solved in a least-squares sense.

Our contact determination algorithm aims to overcome these limitations and is specifically designed to produce robust contact constraints in case of significant interpenetration of detailed surface geometry embedded in coarse simulation geometry. We outline the phases of the algorithm that will be detailed in the next sections:

1. Find the set of intersecting triangle pairs (in section 5.5.1).
2. Compute the set of closed intersection curves $\{\partial\Gamma^i_{AB}\}$ shared by both object surfaces (in section 5.5.2).
3. For each intersection curve, flood inwards on the surface of both objects to find the pair of interior surface regions (Γ^i_A, Γ^i_B) that bound the contact volume region Ω^i_{AB}, and generate a minimal set of contact points (in section 5.5.3).
4. Find contact point correspondences and generate the geometry of all contacts $\{C\} = \{(\boldsymbol{p}_A, \boldsymbol{p}_B, \hat{\boldsymbol{n}}, S)\}$ (in section 5.5.4).

A key aspect of our approach, detailed in section 5.5.4.1, is using the vector area $\boldsymbol{N}_V$ as a smooth estimation of the average contact normal direction $\hat{\boldsymbol{n}}$ for a simple contact volume region Ω^i_{AB} bound by a pair of simply-connected surface regions Γ^i_A, Γ^i_B of the solid object surfaces that intersect along a common boundary curve $\partial\Gamma^i_{AB}$.

5.5.1 Intersecting triangle pairs

We use the nested BVH strategy described in 5.4.4 to find all pairs of intersecting triangles (ITP). Each ITP contains a pair of triangle indices (i_A, i_B), one on each object's *CDG*, the intersection segment $\boldsymbol{q}_0, \boldsymbol{q}_1$, and the specific triangle intersection case (V_AF_B, F_AV_B or E_AE_B), including triangle-intersecting edge indices and their normalized length parameters $\lambda_{0,1} \subseteq [0, 1]$ at the intersection points $\boldsymbol{q}_0, \boldsymbol{q}_1$. Intersection segment endpoints are oriented counter-clockwise on object A and clockwise on object B by convention. The resulting set of ITP contains all segments that define all closed intersection curves between solid objects A and B, that will be expliclitly computed in the next section.

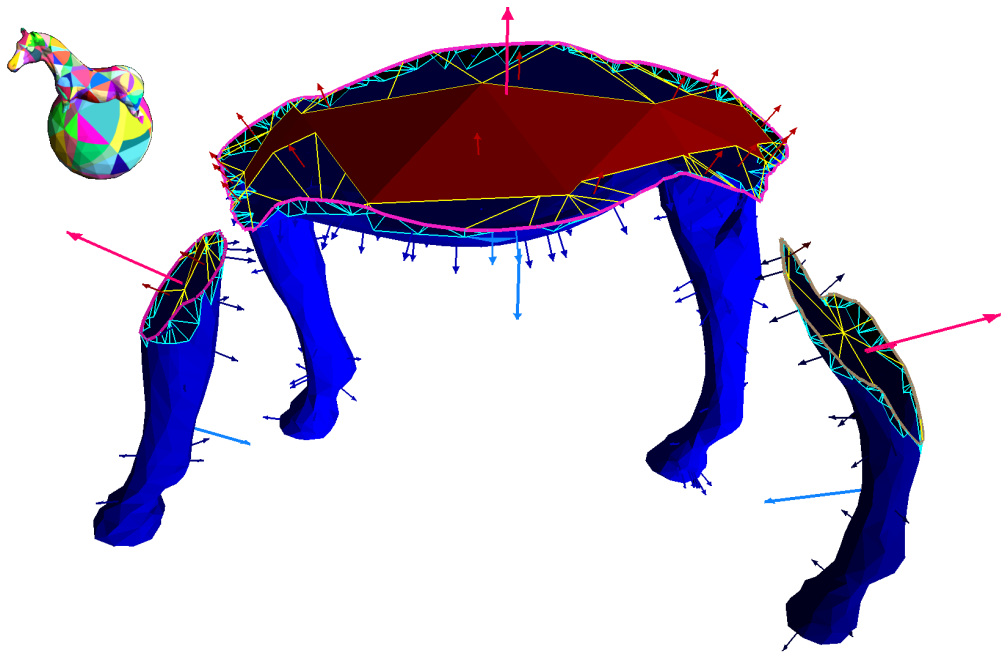

Figure 5.7: A horse model with 5000 triangles intersects a sphere with 1500. and results in 3 disjoint intersection curves $\partial\Gamma^i_{AB}$ that yield as many intersection volumes Ω^i_{AB} bound by 3 pairs of (Γ^i_A, Γ^i_B). We use warm colors for object A and cold colors for object B. Flat-shaded red/blue triangles are completely interior to the other object. Wireframe yellow/cyan polygons are partially clipped triangles. Small arrows show vector area surface normals. Large arrows show vector area contact normals.

5.5.2 Intersection curves

We use a merge-find set data structure directly embedded in the ITP descriptors in order to find the topologically-connected disjoint subsets of ITP that form each closed intersection curve $\partial\Gamma^i_{AB}$. To do so, we use a temporary spatial hashing grid (limited to the intersection volume of the object's global AABB), where all ITP segment endpoints $\boldsymbol{q}_0, \boldsymbol{q}_1$ are first queried for geometric coincidence with previously added endpoints, and added afterwards, which avoids double reporting of symmetric pairs. In order to discard potentially close but unconnected ITP, when a geometric endpoint coincidence within an ϵ radius is found, a topology check requires them to be effectively adjacent on the *CDG* mesh, which can be tested in $\mathcal{O}(1)$ from the ITP information. If successful the ITP are connected through their corresponding shared endpoint and merged into a common parent subset, using path-compression in order to ensure an effective amortized cost $\mathcal{O}(n_{\text{ITP}})$ for the MF-set algorithm [CSRL01].

When all segments have been processed, each MF-set root represents a disjoint intersection curve that is oriented counter-clockwise on object A and clockwise on

object B, as a result of the individual segment orientation convention. The segments in an intersection curve can be iterated in both orientations from the root ITP using their endpoint next/previous connections. Figure 5.7 shows several intersection curves outlined in different colors.

5.5.3 Contact points

In order to find contact points suitable for the definition of contact constraints we will perform a topological flood from each intersection curve $\partial\Gamma^i_{AB}$ inwards on the surface of each object, discover the surface regions (Γ^i_A, Γ^i_B) that bound the contact volume region Ω^i_{AB}, and generate a minimal set of contact points. This process is performed independently on Γ^i_A and Γ^i_B, and will be only detailed for Γ^i_A in the following sections.

5.5.3.1 Vector area surface normals

The vector area of a continuous surface patch Γ is defined as the integral of the unit normal over the entire patch $\boldsymbol{N}_V = \int_\Gamma \hat{\boldsymbol{n}} dA$. The vector area can be used to approximate the normal of a non-planar discrete surface at a given point [CdGDS13]. In the discrete setting, the vector area of a surface patch formed by n_{tri} triangles with vertices $\boldsymbol{v}^k_1, \boldsymbol{v}^k_2, \boldsymbol{v}^k_3$ can be computed as the sum of area-weighted triangle unit normals:

$$\boldsymbol{N}_V = \frac{1}{2}\sum_k^{n_{tri}} (\boldsymbol{v}^k_2 - \boldsymbol{v}^k_1) \times (\boldsymbol{v}^k_3 - \boldsymbol{v}^k_1) \tag{5.3}$$

It can be shown that the vector area only depends on the patch boundary, and can be efficiently computed from its n_{be} counter-clockwise oriented boundary edges $e_k = (\boldsymbol{v}^k_1, \boldsymbol{v}^k_2)$ as:

$$\boldsymbol{N}_V = \frac{1}{2}\sum_k^{n_{be}} \boldsymbol{v}^k_1 \times \boldsymbol{v}^k_2 \tag{5.4}$$

For flat surfaces the vector area computes the polygonal area times the unit normal vector, as expected.

5.5.3.2 Reduced contact points

We do not generate a contact point for each interior vertex or geometric feature. Instead, we restrict contact points $\boldsymbol{p}^j$ to represent simply-connected surface regions inside a single patch P_A in the CDG of a solid object. In a global intersection configuration, a surface patch P_A can have any number of disjoint sub-regions P^j_A individually interior/exterior to the volume of the other solid. Surface patches that are completely interior to the volume of the other object B generate a single contact point, while patches that intersect the surface of the other object generate one contact point for

each disjoint interior sub-region. This strategy ensures that all interior and surface-intersecting patches will contribute individually to the contact geometry, and at the same time, minimizes constraint redundancy on the nodes of the simulation tetrahedrons that contain them. Given a patch sub-region P_A^j we compute its single contact point $\boldsymbol{p}_A^j$, located at the area-weighted centroid, its surface area S^j and its vector area $\boldsymbol{N}_V^j$.

For a completely interior patch we use Eq.5.3 to obtain the vector area. The area and centroid are given by:

$$S^j = \sum_k^{n_{tri}} s^k = \frac{1}{2} \sum_k^{n_{tri}} \|(\boldsymbol{v}_2^k - \boldsymbol{v}_1^k) \times (\boldsymbol{v}_3^k - \boldsymbol{v}_1^k)\| \tag{5.5}$$

$$\boldsymbol{p}^j = \frac{1}{3S^j} \sum_k^{n_{tri}} s^k (\boldsymbol{v}_1^k + \boldsymbol{v}_2^k + \boldsymbol{v}_3^k) \tag{5.6}$$

For an intersecting patch P_A^j there may be 2 kinds of triangles: completely interior and partially clipped. A triangle can be clipped by any number of triangles in the surface of the other object B, resulting in an arbitrary number of planar polygons defined by n_{be} boundary edges $e_k = (\boldsymbol{v}_1^k, \boldsymbol{v}_2^k)$. Their vector area is computed from Eq.5.4 and their area from $s = \|\boldsymbol{N}_V\|$. The centroid of a polygon is:

$$\boldsymbol{c} = \frac{1}{6s} \sum_k^{n_{be}} \|\boldsymbol{v}_1^k \times \boldsymbol{v}_2^k\| (\boldsymbol{v}_1^k + \boldsymbol{v}_2^k) \tag{5.7}$$

Thanks to the additive nature of centroids, areas and vector areas we can process interior and clipped triangles individually, using Eqs.5.5,5.6,5.3 for the former and Eqs.5.7,5.4 for the latter, add all individual contributions and, in the case of the centroid, normalize it with the total accumulated surface area. Surface normals at reduced contact points are shown as short red/blue arrows in Figure 5.7.

5.5.3.3 Clipped triangles

Each $\partial\Gamma_{AB}^i$ is defined by the set of ITP that generate it, which contain all information required to retrieve the set of triangles T_A^m involved, clip them to obtain the planar polygonal sub-regions interior to B, and accumulate their individual contribution to the reduced contact point magnitudes described in section 5.5.3.2. We leverage the orientation conventions for intersection curve segments and for individual triangles in a *CDG* to avoid building the polygonals explicitly. Instead, we only need their constitutive vertices and oriented segments, that can be added in any order in Eqs.5.4,5.7. Clipped triangle polygons are shown as yellow/cyan wireframe in Figure 5.7.

5.5.3.4 Interior triangles

The set of triangles T_A^m that are completely interior to the other object and define a simply-connected surface region Γ_A^i is found using a recursive flood algorithm. The flood is seeded with the interior triangles adjacent to any edge that pierces the other surface, which are available in the ITP of the intersection curve $\partial\Gamma_{AB}^i$.

The flood propagates through triangle neighbour topology available in the *CDG* structure, and stops when there are no more interior triangles adjacent to the flooded region. We completely avoid any expensive triangle-inside-solid tests, instead we test if a candidate triangle adjacent to a flooded interior triangle is involved in any ITP and, if not, accept it as interior. The ITP are hashed and can be searched in expected $\mathcal{O}(1)$. Each discovered interior triangle accumulates its individual contribution to the reduced contact point magnitudes described in section 5.5.3.2. The whole flood process has a computational cost linear in the number of interior triangles. Completely interior flooded region shown as flat shaded red/blue triangles in Figure 5.7.

5.5.3.5 Interior patches

The computational complexity of the interior triangle flood process is output sensitive and asymptotically optimal, as all triangles need to contribute to the reduced contact points. However, it can be very slow for non-shallow interpenetration between highly detailed *CDG* due to the large number of interior triangles to be considered. Fortunately, the partitioned structure of the *CDG* allows us to flood at the patch level, efficiently covering large areas of the Γ_A^i without accessing individual triangles. To do so, we use the patch-level topology precomputed in the *CDG*. As with triangles, patches are classified as interior if they are adjacent to an interior patch or triangle and do not contain any non-interior triangle. This can be efficiently computed by setting a non-interior flag on the parent patch of any triangle involved in an ITP.

As mentioned before, all individual interior triangles must contribute to the reduced contact points. We can benefit again from the additive definition of the centroid and the vector area to compute the aggregate contribution of all triangles in a barycentrically deformed patch in $\mathcal{O}(1)$ from their precomputed undeformed values stored in the *CDG*. The centroid is transformed with $\mathcal{G}_m^s$, the vector area transformation is detailed in the appendix A.4. This optimization is not possible for the deformed patch surface area, which is approximated using the undeformed value.

5.5.4 Contact point correspondences

When all reduced contact points on all Γ_A^i, Γ_B^i have been independently generated we need to determine their *correspondences* on the opposite surface Γ_B, Γ_A to generate the individual contact constraints $\{C\} = \{(\boldsymbol{p}_A, \boldsymbol{p}_B, \hat{\boldsymbol{n}}, S)\}$. Focusing on object A, for

each $\boldsymbol{p}_A^j$ we need to find an opposite point $\boldsymbol{p}_B(\boldsymbol{p}_A^j)$, a normal direction $\hat{\boldsymbol{n}}(\boldsymbol{p}_A^j, \boldsymbol{p}_B^j)$ and a contact area $S(\boldsymbol{p}_A^j)$.

In proximity-based contact determination the opposite point is the closest point on Γ_B, and the normal is aligned with the direction $\boldsymbol{p}_B - \boldsymbol{p}_A$. However, this strategy is not adequate for non-shallow interpenetrations. An alternative ray-traced collision detection strategy was proposed in [HFR08], where correspondences are found by tracing a ray from $\boldsymbol{p}_A^j$ in a direction opposite to the surface normal at the point. This approach only works reliably for convex contact volumes and is therefore severely limited. A parametric correspondence method for cloth simulation was introduced in [WLG06], but is not directly applicable to the surface of solid objects with arbitrary genus.

5.5.4.1 Vector area contact normals

We experimented with several alternative methods and found that performing a raycast from each $\boldsymbol{p}_A^j$ along a single global normal $\hat{\boldsymbol{n}}_V^i$ for each contact region Ω_{AB}^i, computed from the normalized vector area of the intersection curve $\partial\Gamma_{AB}^i$, yields very good results for both shallow and moderately deep interpenetrations and overcomes the limitations of previous approaches. Each ray is cast from a small offset behind the actual $\boldsymbol{p}_A^j$ to avoid geometrical coincidences in case of near-coplanar shallow contacts. Raycast queries on potentially deformed solids are accelerated using their nested BVH (section 5.4.4). A reduced contact point $\boldsymbol{p}_A^j$ only generates a contact constraint if its vector area normal $\hat{\boldsymbol{n}}_A^j$ and the correspondent point $\boldsymbol{p}_B(\boldsymbol{p}_A^j)$ found by the raycast query satisfy the contact conditions $(\hat{\boldsymbol{n}}_V^i)^T \hat{\boldsymbol{n}}_A^j < 0$ and $(\hat{\boldsymbol{n}}_V^i)^T \hat{\boldsymbol{n}}_B > 0$.

The main benefits of using $\hat{\boldsymbol{n}}_V^i$ as a single global normal for a whole contact region Ω_{AB}^i are:

- Represents an area weighted average of the surface normals and therefore models the contact constraint on the whole contact surface in an average sense.
- Is independent of the detail of the interior surfaces, that may be very complex (Figure 5.7), and filters out high frequency surface detail that may produce erratic, globally inconsistent local contact normals (Figure 5.6).

The vector area contact normal $\hat{\boldsymbol{n}}_V^i$ has some relevant properties:

(a) Is exact if either Γ_A^i or Γ_B^i are flat, regardless of the complexity of the other surface.

(b) Is exact for a single vertex-face or edge-edge contact, as proved in Appendix A.3).

(c) Converges to the exact contact normal as the perimeter of $\partial\Gamma_{AB}^i$ tends to zero.

(d) Is parallel to the volume gradient of the contact region [CdGDS13].

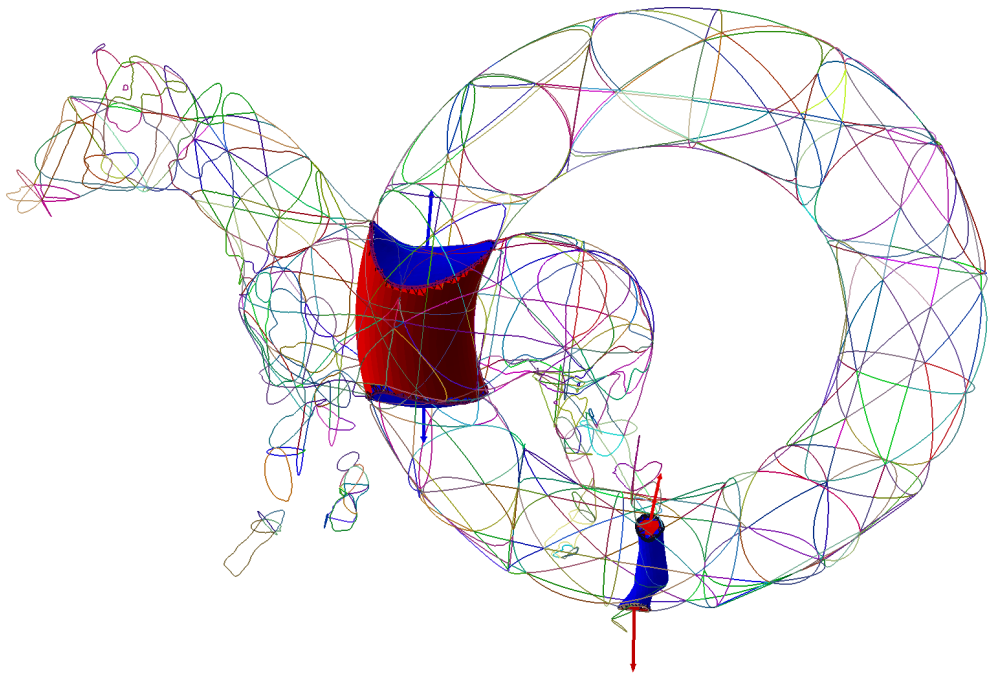

Figure 5.8: The intersection of a horse and a torus results in two tube-shaped intersection volumes Ω^1_{AB}, Ω^2_{AB}, each one bound by two boundary surface regions on one object and one on the other. Patch boundaries shown in wireframe.

(e) Only depends on $\partial\Gamma^i_{AB}$ and changes with the same continuity.

Properties (a) and (b) guarantee exact results in simple contact configurations, including sharp features. Property (c) implies that the contact normal approximation is self-correcting if the perimeter of $\partial\Gamma^i_{AB}$ decreases, as expected, when contact treatment pushes the surfaces apart along its direction. Property (d) implies that $\hat{\boldsymbol{n}}^i_V$ is the best single direction to push objects apart in order to minimize the intersection volume. Finally, property (e) guarantees that smooth simulation trajectories result in smooth changes in the contact normal.

For large interpenetrations the global normal hit point from $\boldsymbol{p}^i_A$ may end outside A. To avoid it, we perform an additional raycast on Γ_A from $\boldsymbol{p}^i_A$ and, if a closer hit is found, we discard $\boldsymbol{p}^i_A$. This is only necessary when $\boldsymbol{p}_B$ is beyond a threshold distance corresponding to the object's minimum thickness and is rarely necessary for shallow and moderately deep contacts.

5.5.5 Complex intersection topology

Non-simple volumes Ω^i_{AB} may result when considering the unrestricted intersection of solid objects A, B represented by closed surfaces $\Gamma_{\mathrm{A}}, \Gamma_{\mathrm{B}}$ with arbitrary genus. The volume Ω^i_{AB} may be bound by any number of surface regions in one object, and may connect disjoint intersection curves on the other, as shown in Figure 5.8. This scenario is extremely challenging for any local contact determination method. Our flood based algorithm can detect these problematic cases and avoid re-flooding the same Γ^i_{A} from its multiple $\partial\Gamma^i_{\mathrm{A}}$ seeds. However, the computed global normal and reduced contact point correspondences will only be consistent if the individual global normals of all involved $\partial\Gamma^i_{\mathrm{A}}$ are roughly aligned.

5.5.6 Differences with volumetric contact determination

The proposed flood-based method shares certain similarities with GPU-based volumetric contact determination [AFC+10]. Both approaches essentially build an implicit representation of each independent intersection volume and perform global analysis on its geometry. Contact normals are only defined in an average sense and depend on the whole intersection volume. However, significant differences exist:

- Our flood-based method generates shape-dependent reduced contact points that affect all DOF of the elements involved. On the other side, GPU-based methods generate contact points at a given spatial resolution, with no shape dependency, and may underresolve thin features.

- Our vector-area global contact normal is parallel to the global gradient of the intersection volume [CdGDS13]. Similarly, volumetric methods define local contact point normals as the local gradients of the global intersection volume, which induces local contact response that works to minimize it. This local response may cause collapse of intersection regions with genus > 1, driving their volume to 0 without actually resolving the contact. In addition, closed holes (eg: bubbles) inside intersection volumes would grow due to local volume minimization.

- Computationally, GPU-based techniques leverage the immense computational power of modern GPUs to execute essentially brute-force "embarrasingly parallel" algorithms fast. While this may be acceptable in dedicated virtual training applications, it is not practical for high-definition videogames, where most GPU power is required for visualization.

	VIZ.T	SIM.E	CDG.E	CDG.P	CDG.T	CDG.V
Armadillo1K	1K	452	440	522	6K	6.5K
Armadillo5K	5K	465	449	533	15K	14K
Armadillo10K	10K	472	456	552	24K	20K
Armadillo100K	100K	483	467	563	130K	84K
Dragon1K	1K	411	411	557	6K	7K
Dragon5K	5K	411	411	590	16K	15K
Dragon10K	10K	408	408	586	25K	21K
Dragon100K	100K	411	411	598	130K	85K
Horse1K	1K	283	281	336	5K	5K
Horse5K	5K	287	285	356	13K	11K
Horse10K	10K	287	285	354	20K	16K
Horse100K	100K	287	285	366	117K	73K

Table 5.1: Model sizes for the visualization (VIZ), simulation (SIM) and collision detection (CDG) geometry, specified in (T)riangles, (E)lements, (P)atches and (V)ertices.

5.6 Results

We implemented the contact determination algorithm as part of the interactive corotational FEM simulation scheme described in previous chapters. All examples use the same soft material properties (Young modulus 1000, Poisson ratio 0.25) and a constant simulation step $\Delta t = 1/60$s as common in videogames. The dynamics equations are integrated using the Backwards Euler method with an iterative MINRES solver limited to 10 iterations per timestep. Contacts are solved using the approach described in Section 3.4.2.2 with a relaxation coefficient $k_r = 0.1$ to smoothly correct interpenetrations once detected. We did not use any prediction-correction step or pre-interpenetration contact detection strategy in order to stress the post-interpenetration recovery benefits of the proposed algorithm. The dynamics simulation cost was below 5ms per timestep in all tests.

We simulated different contact-heavy scenes with models of varying detail (Table 5.1) to determine the behaviour of our contact determination algorithm. The method performed well in general, successfully recovering shallow and moderately deep interpenetrations in a few milliseconds. Significantly deep interpenetrations are resolved if the contact volume remains topologically simple or, at least, its multiple intersection curves yield compatible normals. However, in extreme situations (Figure 5.8) the simulation can become unrealistic due to mutually cancelling contact reactions.

In Figure 5.9 a dynamic armadillo collides with a static nest after 2 seconds of free fall, and interpenetrates it at high speed. The simulation runs at constant 60fps. Contact determination cost peaks at 11.2ms, with up to 800 ITP and 100 reduced contact points distributed in a maximim of 19 disjoint intersection curves. The high number of intersection curves is caused by the roughness of the surfaces in contact. High-

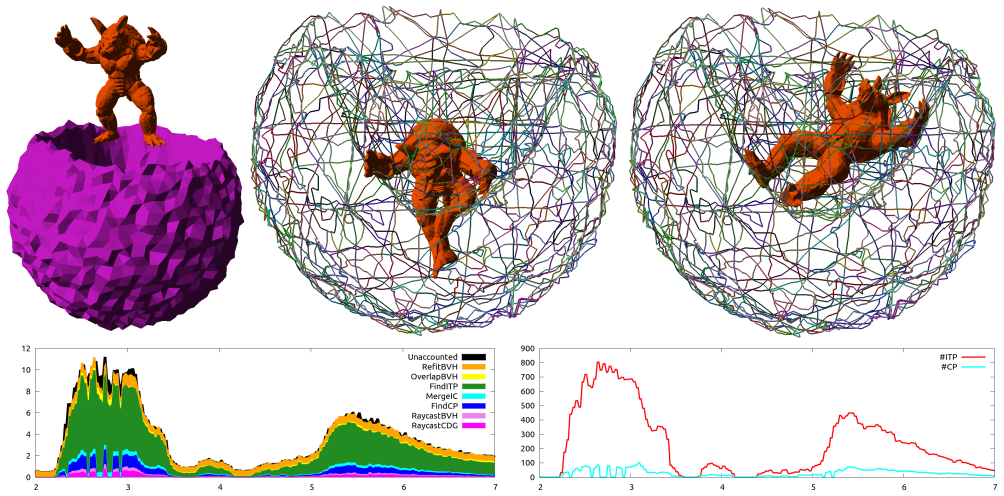

Figure 5.9: Top row: Armadillo5K falls on a non-smooth nest with 5K triangles at a high speed, causing severe interpenetration (mid) that is robustly corrected in less than a second using the contact determination algorithm (right). Bottom row: Stacked CPU cost for the different phases of the contact detection algorithm (left) and contact triangle/point count (right). Plots start on impact $t \approx 2$s.

speed collision scenarios greatly benefit from a predictive contact detection approach that allows early contact treatment, specially if computationally intensive prediction-correction iterations are performed. In their absence, or as a fallback strategy when interpenetration cannot be avoided, our contact determination algorithm successfully corrects it and returns the armadillo to non-interpenetration safety in less than a second. The long recovery time is due to the soft material properties used, that delay contact reaction propagation to the upper body. Meanwhile, the feet remain stable at a small depth under the nest surface.

In a different experiment the armadillo tries to ride a horse (Figure 5.11). From an initial configuration 0.5m above the horse with both legs intersecting it sides, the armadillo falls under gravity acceleration and spends 5 seconds in frictional contact with the horse surface (up to 9 contact regions). The simulation was run with 3 exponential levels of geometric detail (10^3, 10^4, 10^5 triangles per model) and the same level of dynamic detail (≈ 450 dynamic tetrahedrons). Node trajectories differ but are qualitatively similar. The plots in Figure 5.10 show the stacked CPU time spent in each phase of the contact determination algorithm during 5 seconds of simulation time.

Benchmark results in Figures 5.9 and 5.10 show that the cost of finding the ITP (green) largely dominates across all levels of detail. This includes all b-DOP-Tree and exact triangle-triangle intersection tests. Merging the ITP into intersection curves (cyan), flooding the surfaces to discover the reduced contact points (blue) and ray-

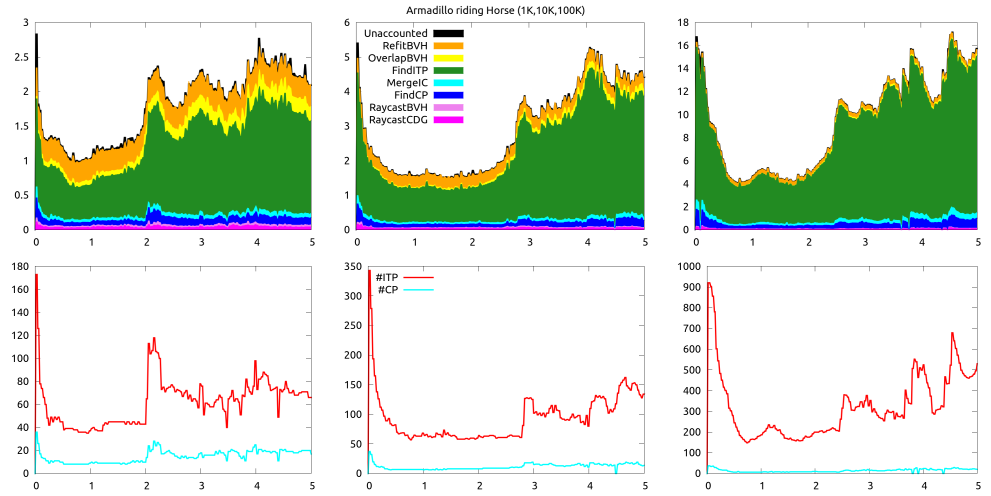

Figure 5.10: Results of the horse-riding armadillo simulation with 1K (left),10K (mid) and 100K (right) models for both objects. Top: Stacked CPU cost for the different phases of the contact detection algorithm. Bottom: Number of intersecting triangle pairs (ITP) and reduced contact points (CP).

casting along the vector area global normals to find the contact point correspondences (magenta) all require a much smaller fraction of the computation accross all detail levels. The cost of refitting (orange) and testing (yellow and pink) the coarse BVH is constant for all levels of detail, as it only depends on the common dynamic mesh.

The bottom row in Figure 5.10 shows the total number of ITP and reduced contact points (CP) found for each level of detail. The number of ITP grows sublinearly with the surface resolution. The number of CP remains very similar at all levels of detail, as it strongly depends on the number of patches in the *CDG*, in the same order of magnitude as the number of tetrahedrons in the dynamics mesh (Table 5.1). There is a clear correlation between the total cost of contact determination and the number of ITP detected (red), suggesting a strongly output-sensitive performance. We performed statistical analysis on the CPU and contact complexity time series and found strong evidence that MergeIC (MF-set) and FindCP (flood) are linear on the number of ITP, with correlation coefficients $r = 0.97$ and $r = 0.92$ respectively. FindITP cost does not only depend on the number of ITP, as it also increases due to non-intersecting close proximity. Unaccounted time (black) in the plots represents computational overhead external to the core algorithm, mostly related to dynamic memory management and sample collection, and represents a tiny fraction of the total cost.

The effect of the different optimizations detailed throughout the paper is analyzed in Table 5.2 for the static configuration shown in Figure 5.11 that requires 30ms for contact determination. This significant intepenetration depth is never reached during

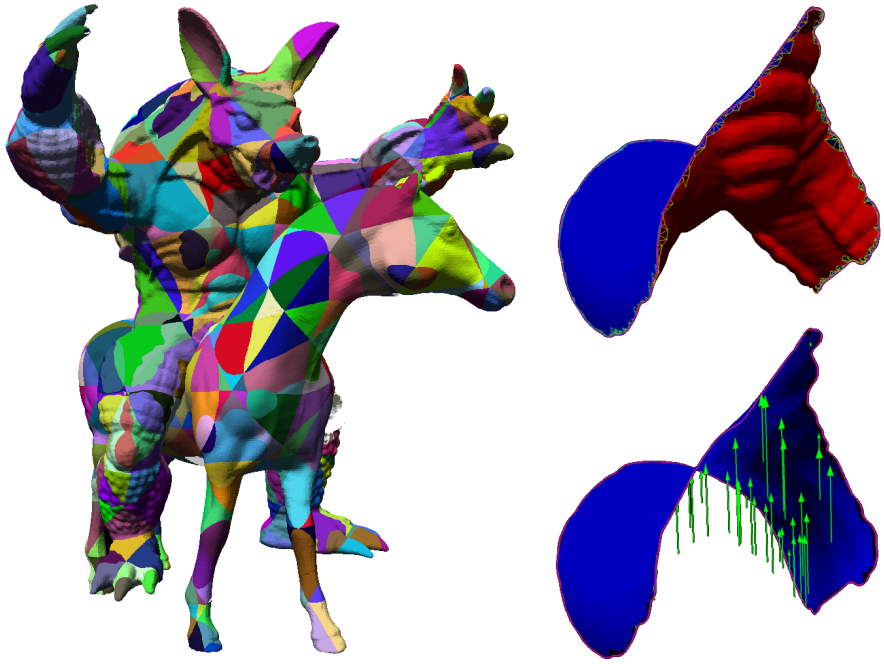

Figure 5.11: Armadillo100K intersecting Horse100K (left). Intersection boundary surfaces (top-right) and reduced contact point normals (bottom-right)

	Total	BVH	FindITP	MergeIC	FindCP	RayCDG
All	30	1	17	1	9	1.8
No b-DOP-T	1753	1	1705	1	9	32
$T_{\text{int}} = 1$	33	1	20	1	9	1.8
No P Flood	34	1	17	1	13	1.8

Table 5.2: Columns: Timings in milliseconds for the phases of the CD algorithm. Rows: Effect of the various optimizations in the configuration shown in Figure 5.11. The CD algorithm finds in a single intersection curve formed by 1296 ITP that bounds a total of 20062 triangles in 44 internal patches and 12 intersecting patches, and generates 56 reduced contact points. BVH column includes coarse BVH refit, intersection and raycast. The first row includes all optimizations. No b-DOP-T performs brute-force intersection and raycast tests on all *CDG* triangles in all elements overlapped in the coarse BVH. $T_{\text{int}} = 1$ disables the dynamic "leaf" strategy in b-DOP-Tree intersection tests. No P Flood disables *CDG* patch-level navigation during flood that forces it to consider 7483 additional triangles individually and avoids $\mathcal{O}(1)$ transformation of patch centroids and vector areas.

the simulation benchmarks in Figure 5.10, that peak at 16ms for the same level of detail.

The overall results are qualitatively and quantitatively satisfactory. Moderately intersecting models with up to 100K visualization triangles can be processed in less than 1/60s in our modest test hardware (Intel Core i5-2500K at 3.3 GHz), with an output-sensitive computational cost that grows with the geometric complexity of the intersection region. Our single-core C++11 implementation is reasonably fast but still leaves much room for low level optimization, such as avoiding all dynamic memory allocation (std::vector, std::unordered_map) and using SIMD for math-heavy geometric transformations and tests. Our basic coarse BVH refit would benefit from a more efficient alternative. Different phases of the algorithm could be performed in parallel on both objects, and aligned raycasts along the global normal could be optimized.

For highly detailed surfaces the results justify the increased geometric feature count caused by triangle mesh partitioning along tetrahedron boundaries. This increased complexity could be reduced if the CD geometry is allowed to diverge from the visualization geometry, using per-patch mesh simplification while keeping inter-patch boundary edges unmodified. In a production environment this process could be artist-guided in order to ensure optimal accuracy in the most relevant parts of a model.

5.7 Discussion

We proposed the use of a specific collision detection geometry with a partitioned surface representation decoupled from the visualization geometry and barycentrically embedded in a coarse simulation tetrahedral mesh. The partitioned representation enables several algorithmic and low level optimizations that yield an efficient scheme for highly detailed collision geometry.

Our contact determination approach uses an averaged definition of the contact normal that is robust in presence of high frequency surface detail and severe interpenetrations, both source of problems in other methods, and can be considered *global* for topologically simple intersection regions. The proposed scheme can be used in isolation if objects are allowed to interpenetrate, or, more efficiently, in conjunction with an exterior proximity-based approach, acting as a robust fallback method in case of unavoidable intersection. The optimized *CDG* representation could be used for proximity-based detection in non-intersecting areas in close proximity to implement a hybrid proximity/intersection contact determination method.

As a limitation, our approach requires surface intersection and cannot detect completely contained objects. We will address this issue using the internal tetrahedron volume. Additionally, we cannot reliably solve general, topologically complex intersections, a limitation shared by previous works. This is an avenue for future research. We also plan to deal with self-collision using the *CDG* acceleration structures and the $\mathcal{O}(1)$ deformed patch vector area normals.

CHAPTER 6

Conclusions and Future Work

6.1 Summary

This thesis is the result of our work on the interactive simulation of elastically deformable solids for videogames and related application areas defined by strict robustness and efficiency requirements but weak physical accuracy demands. Our research focused on the solution of practical problems encountered during the development of a state-of-the-art corotational linear FEM simulation library, and resulted in new methods to solve them. Our main contributions are summarized next

- In Chapter 3 we described a fast and robust simulation scheme for deformable solids modelled using corotational linear FEM, and analyzed the multiple trade-offs between accuracy and computational cost. We proposed a new stress differential approximation that improves the accuracy of inexact implicit integration for large deformations and closely matches the fully-implicit solution with minor computational overhead.

- In Chapter 4 we dealt with degenerate element treatment, a critical problem in FEM-based simulation. Building on our previously published work [CFS14], we presented an improved *degeneration-aware polar decomposition* method that guarantees the smothness of extracted rotations and benefits numerical solver convergence.

- In Chapter 5 contact determination for intersecting deformable solids was considered, and we presented an end-to-end solution that includes efficient automatic

tetrahedralization, precomputation of a dedicated geometry representation with an embedded memory-efficient BVH, and a flood-based contact determination algorithm that uses a new definition of contact normals. A long paper describing this work has been accepted at *Motion in Games 2015*.

6.2 Conclusions

Simulation of deformable solids in videogames has been very limited so far. Our hypothesis is that computational cost and robustness issues stop its widespread use. The current trend in the field focuses on Position Based Dynamics (PBD) methods. This model is very popular for cloth and hair simulation in videogames for its simplicity, efficiency and stability, and can be applied to deformable solids using either a geometrically motivated shape-matching constitutive model [BMOT13], or a strain-based approach that uses elastic energy as an energy constraint to be minimized [BKCW14, MCKM14]. Neither option is consistently derived from continuum mechanics, and both use unphysical material parameters and yield discretization-dependent results. The recently introduced projective dynamics approach [BML+14] combines the simplicity and efficiency of PBD with a continuum mechanics foundation that reduces discretization-dependence of the simulation results, but still relies on unphysical material parameters. On the other hand, the Finite Element Method applied to continuum mechanics is generally considered too expensive, specially in case of higher order elements and nonlinear materials. Consistent approximations are possible within the FEM framework, and the corotational linear constitutive model offers a suitable compromise between accuracy, efficiency and robustness. We hope that our contributions will help physically-consistent FEM schemes close the gap with more pragmatic alternatives like PBD.

The decision to use coarse simplex meshes for simulation and barycentrically embedded detailed geometry for visualization and contact allowed us to achieve fast update rates with moderate CPU cost while preserving visual detail. Non-conforming, strictly bounding tetrahedral meshes have proved to be an efficient choice for CPU-limited interactive simulation. This approach allows fast GPU-based surface deformation for visualization, and is directly compatible with the standard approach for character rendering using kinematic skeletal animation with linear-blend skinning available in virtually all videogame engines.

Contact determination is an essential part of interactive simulation, and represents a large fraction of the computational effort required. Proximity-based methods are generally preferred, but cannot strictly avoid solid geometry intersection. Robust and efficient treatment of intersecting contact for highly-detailed geometry is a problem scarcely addressed in interactive simulation research. The state-of-the-art is dominated

by GPU-based volumetric methods using LDI or sphere-based geometry representations that lead to uniform brute-force computations, but are limited in both geometric resolution and topologic complexity. Flood-based methods are better suited to CPU execution due to their complex combinatorial nature, but allow higher-level reasoning on contact regions, essential for global contact determination and treatment. Our proposed flood-based method is fast for shallow interpenetrations and offers improved robustness for moderately deep intersections.

6.3 Future work

Corotational linear FEM with implicit Euler integration has numerous limitations that we'd like to overcome in future work:

- Energy dissipation: Implicit Euler integration provided us with very good stability for large timesteps at the cost of energy dissipation. In future work we'll experiment with more accurate methods.
- Stiff materials and strain limiting: At present we're limited to soft, moderately compressible materials, it would be interesting to incorporate ideas from constraint-based elasticity [SLM06, TNGF15] to support stiff materials and compliant strain limits.
- Nonlinear materials: Linear materials offer very limited realism in presence of large stretches and extreme compression. We performed some tests using the nonlinear Corrected Corotated Model (CCM) that includes incompressibility terms and can also benefit from DAPD rotations. The results were satisfactory but impractical for real-time purposes, requiring several Newton-Raphson iterations to converge robustly. We plan to experiment with CCM approximations that allow for faster, unconditional convergence.
- Nondegeneration constraints and DAPD drift-control: We performed some preliminary experiments on degeneration prevention through reaction constraints embedded in the linear system solver using the continuous-time DAPD degeneration direction $\hat{\boldsymbol{d}}_c$ as the constraint gradient, and DAPD forces for constraint stabilization. This prevented inversion even for extreme gravity, but required several Newton-Raphson iterations to become stable, and eventually diverged. Further research could help reduce computational cost and ensure convergence.
- Fracture: Our partitioned surface representation is suitable for fast pre-scored fracture along element boundaries, that will be considered in the near future.
- Plasticity: Although not reported in this document, we have implemented basic plasticity using additive strain decomposition. It would be interesting to study

the interplay between plasticity and element degeneration, and compare the behaviour of additive and multiplicative strain decompositions on degeneration.

Regarding geometry representation and contact determination, several avenues for future work are worth exploring:

- Tetrahedralization: Our simplistic mesh-generation method in Section 5.3.1 could be improved with additional optimization criteria, apart from the ODT energy, and a more complex global optimization scheme based in random node perturbation, at a higher precomputation cost.
- Hybrid proximity/intersection contact determination: The application of our dedicated contact geometry representation (CDG) for proximity-based contact determination will hopefully result in a hybrid scheme that combines the best of both approaches, and will be also used for self-collision detection.
- Parallelization and GPU-acceleration: Flood-based contact determination could be accelerated by parallelization and delegation of specific subtasks to the GPU, such as massive aligned raytracing.
- Solid object untangling: Solving topologically complex intersection configurations is an open problem. We plan to address it by extending our efficient flood-based scheme using global topologic analysis of the intersection regions.

Appendix

A.1 Derivation of $\mathcal{P}_D$ and $\delta\mathcal{P}_D$

In this appendix we discuss the exact derivation of $\mathcal{P}_D$ and $\delta\mathcal{P}_D$ from the degeneration-aware corotational stiffness tensor $\epsilon_D = \frac{1}{2}(\hat{\mathcal{S}}^T + \hat{\mathcal{S}}) - \mathcal{I}$. The matrix $\hat{\mathcal{S}} = \bar{\mathcal{R}}^T\mathcal{F}$ is asymmetric in general, and results from the rotation $\bar{\mathcal{R}}$ obtained by polar decomposition of a modified deformation $\bar{\mathcal{F}} = \bar{\mathcal{D}}_s\mathcal{D}_m^{-1}$, where $\bar{\mathcal{D}}_s$ is computed as described in Section 4.4.2. The computation can be simply stated as:

$$\bar{\mathcal{D}}_s = \mathcal{D}_s + \bar{\mathcal{U}} \tag{A.1}$$

$$\bar{\mathcal{F}} = \mathcal{F} + \bar{\mathcal{U}}\mathcal{D}_m^{-1} \tag{A.2}$$

$$\delta\bar{\mathcal{F}} = \delta\mathcal{F} + \delta\bar{\mathcal{U}}\mathcal{D}_m^{-1} \tag{A.3}$$

where $\bar{\mathcal{U}}$ is results from the case-dependent displacements applied by the DAPD scheme and is 0 for undegenerate configurations. We define the intermediate matrix $\mathcal{A}$ and its differential

$$\mathcal{A} = \hat{\mathcal{S}}^T + \hat{\mathcal{S}}, \quad \delta\mathcal{A} = \delta\hat{\mathcal{S}}^T + \delta\hat{\mathcal{S}} \tag{A.4}$$

and rewrite the strain tensor and its differential as

$$\epsilon_D = \frac{1}{2}\mathcal{A} - \mathcal{I}, \quad \delta\epsilon_D = \frac{1}{2}\delta\mathcal{A} \tag{A.5}$$

We will use the following properties of the trace

$$\begin{aligned}
\operatorname{tr}(\alpha\mathcal{A}) &= \alpha\operatorname{tr}(\mathcal{A}) \\
\operatorname{tr}(\mathcal{A}) &= \operatorname{tr}(\mathcal{A}^T) \\
\operatorname{tr}(\mathcal{A}\mathcal{B}) &= \operatorname{tr}(\mathcal{B}\mathcal{A}) \\
\operatorname{tr}(\mathcal{A}^T\mathcal{B}) &= \operatorname{tr}(\mathcal{A}\mathcal{B}^T) \\
\operatorname{tr}(\mathcal{A}+\mathcal{A}^T) &= 2\operatorname{tr}(\mathcal{A}) \\
\operatorname{tr}(\mathcal{A}(\mathcal{B}+\mathcal{B}^T)) &= 2\operatorname{tr}(\mathcal{A}\mathcal{B}) \quad \text{if} \quad \mathcal{A}=\mathcal{A}^T \\
\operatorname{tr}(\mathcal{A}^T\mathcal{A}) &= \|\mathcal{A}\|_F^2 \\
\operatorname{tr}(\mathcal{A}\mathcal{B}) &= \mathcal{A}^T : \mathcal{B}
\end{aligned}$$

and rewrite the elastic energy Ψ_D as

$$\Psi_D = \mu\|\epsilon_D\|_F^2 + \frac{\lambda}{2}\operatorname{tr}^2(\epsilon_D) \tag{A.6}$$

$$= \underbrace{\mu\operatorname{tr}(\epsilon_D^T\epsilon_D)}_{\Psi_\mu} + \underbrace{\frac{\lambda}{2}\operatorname{tr}^2(\epsilon_D)}_{\Psi_\lambda} \tag{A.7}$$

with

$$\Psi_\mu = \mu\operatorname{tr}\left[(\frac{1}{2}\mathcal{A}-\mathcal{I})^T(\frac{1}{2}\mathcal{A}-\mathcal{I})\right] \tag{A.8}$$

$$= \mu\operatorname{tr}\left[\frac{1}{4}\mathcal{A}^T\mathcal{A} - \frac{1}{2}\mathcal{A}^T\mathcal{I} - \frac{1}{2}\mathcal{I}^T\mathcal{A} + \mathcal{I}^T\mathcal{I}\right] \tag{A.9}$$

$$= \mu\left[\frac{1}{4}\operatorname{tr}(\mathcal{A}^T\mathcal{A}) - \operatorname{tr}(\mathcal{A}) + \operatorname{tr}(\mathcal{I})\right] \tag{A.10}$$

In order to compute $\mathcal{P}_D = \frac{\partial\Psi_D}{\partial\mathcal{F}}$ we follow a derivation similar to the Technical Notes in [MZS+11a]. We begin with

$$\delta\Psi_D = \frac{\partial\Psi_D}{\partial\mathcal{F}} : \delta\mathcal{F} \tag{A.11}$$

$$= (\frac{\partial\Psi_\mu}{\partial\mathcal{F}} + \frac{\partial\Psi_\lambda}{\partial\mathcal{F}}) : \delta\mathcal{F} \tag{A.12}$$

$$= \delta\Psi_\mu + \delta\Psi_\lambda \tag{A.13}$$

and compute each differential separatedly

$$\delta\Psi_\lambda = 2\frac{\lambda}{2}\operatorname{tr}(\epsilon_D)\operatorname{tr}(\delta\epsilon_D) \tag{A.14}$$

$$= \frac{\lambda}{2}\operatorname{tr}(\epsilon_D)\operatorname{tr}(\delta\mathcal{A}) \tag{A.15}$$

and

$$\delta\Psi_\mu = \mu\left[\frac{1}{4}\operatorname{tr}(\delta\mathcal{A}^T\mathcal{A} + \mathcal{A}^T\delta\mathcal{A}) - \operatorname{tr}(\delta\mathcal{A})\right] \tag{A.16}$$

$$= \mu\left[\frac{1}{2}\operatorname{tr}(\mathcal{A}^T\delta\mathcal{A}) - \operatorname{tr}(\delta\mathcal{A})\right] \tag{A.17}$$

$$= \mu\operatorname{tr}\left((\frac{1}{2}\mathcal{A}^T - \mathcal{I})\delta\mathcal{A}\right) \tag{A.18}$$

$$= \mu\operatorname{tr}(\epsilon_D\delta\mathcal{A}) \tag{A.19}$$

At this point, for undegenerate configurations with $\bar{\mathcal{R}} = \mathcal{R}$, $\bar{\mathcal{F}} = \mathcal{F}$ and $\hat{\mathcal{S}} = \mathcal{S} = \mathcal{S}^T$ it can be shown that

$$\operatorname{tr}(\delta\mathcal{A}) = 2\operatorname{tr}(\delta\mathcal{S}) = 2\operatorname{tr}(\mathcal{R}^T\delta\mathcal{F} + \delta\mathcal{R}^T\mathcal{F}) = 2\operatorname{tr}(\mathcal{R}^T\delta\mathcal{F}) \tag{A.20}$$

thanks to the cancellation

$$\operatorname{tr}(\delta\mathcal{R}^T\mathcal{F}) = \operatorname{tr}(\delta\mathcal{R}^T\mathcal{R}\mathcal{S}) = 0 \tag{A.21}$$

because $\delta\mathcal{R}^T\mathcal{R}$ is skew-symmetric and $\mathcal{S}$ is symmetric (see [MZS+11a] and its accompaining Technical Notes). Therefore, the undegenerate energy differential would be

$$\delta\Psi'_D = 2\mu\operatorname{tr}(\epsilon_D\delta\mathcal{S}) + \lambda\operatorname{tr}(\epsilon_D)\operatorname{tr}(\delta\mathcal{S}) \tag{A.22}$$

$$= 2\mu\operatorname{tr}(\epsilon_D\mathcal{R}^T\delta\mathcal{F}) + \lambda\operatorname{tr}(\epsilon_D)\operatorname{tr}(\mathcal{R}^T\delta\mathcal{F}) \tag{A.23}$$

$$= \mathcal{R}\big[2\mu\epsilon_D + \lambda\operatorname{tr}(\epsilon_D)\mathcal{I}\big] : \delta\mathcal{F} \tag{A.24}$$

and result in the stress

$$\mathcal{P}'_D = \mathcal{R}\big[2\mu\epsilon_D + \lambda\operatorname{tr}(\epsilon_D)\mathcal{I}\big] \tag{A.25}$$

However, in degenerate configurations with $\bar{\mathcal{R}} \neq \mathcal{R}$ and $\bar{\mathcal{F}} \neq \mathcal{F}$ the matrix $\hat{\mathcal{S}} \neq \mathcal{S}$ is not symmetric, and Equation (A.20) does not hold. Instead, we need to consider all terms in Equation (A.4)

$$\delta\hat{\mathcal{S}} = \delta\bar{\mathcal{R}}^T\mathcal{F} + \bar{\mathcal{R}}^T\delta\mathcal{F} \tag{A.26}$$

$$\delta\mathcal{A} = \delta\hat{\mathcal{S}}^T + \delta\hat{\mathcal{S}} \tag{A.27}$$

$$= (\mathcal{F}^T\delta\bar{\mathcal{R}} + \delta\mathcal{F}^T\bar{\mathcal{R}}) + (\delta\bar{\mathcal{R}}^T\mathcal{F} + \bar{\mathcal{R}}^T\mathcal{F}) \tag{A.28}$$

$$= \underbrace{\bar{\mathcal{R}}^T\delta\mathcal{F} + \delta\mathcal{F}^T\bar{\mathcal{R}}}_{\delta\mathcal{A}_{\mathcal{F}}} + \underbrace{\mathcal{F}^T\delta\bar{\mathcal{R}} + \delta\bar{\mathcal{R}}^T\mathcal{F}}_{\delta\mathcal{A}_{\bar{\mathcal{R}}}} \tag{A.29}$$

We can now split $\delta\Psi$ into $\delta\mathcal{F}$ and $\delta\bar{\mathcal{R}}$ parts:

$$\begin{aligned}
\delta\Psi &= \delta\Psi_\mu + \delta\Psi_\lambda && \text{(A.30)}\\
&= \mu\,\mathrm{tr}(\epsilon_D\delta\mathcal{A}) + \frac{\lambda}{2}\,\mathrm{tr}(\epsilon_D)\,\mathrm{tr}(\delta\mathcal{A}) && \text{(A.31)}\\
&= \mu\,\mathrm{tr}(\epsilon_D(\delta\mathcal{A}_\mathcal{F} + \delta\mathcal{A}_{\bar{\mathcal{R}}})) + \frac{\lambda}{2}\,\mathrm{tr}(\epsilon_D)\,\mathrm{tr}(\delta\mathcal{A}_\mathcal{F} + \delta\mathcal{A}_{\bar{\mathcal{R}}}) && \text{(A.32)}\\
&= \underbrace{\mu\,\mathrm{tr}(\epsilon_D\delta\mathcal{A}_\mathcal{F}) + \frac{\lambda}{2}\,\mathrm{tr}(\epsilon_D)\,\mathrm{tr}(\delta\mathcal{A}_\mathcal{F})}_{\delta\Psi_\mathcal{F}} + \underbrace{\mu\,\mathrm{tr}(\epsilon_D\delta\mathcal{A}_{\bar{\mathcal{R}}}) + \frac{\lambda}{2}\,\mathrm{tr}(\epsilon_D)\,\mathrm{tr}(\delta\mathcal{A}_{\bar{\mathcal{R}}})}_{\delta\Psi_{\bar{\mathcal{R}}}} && \text{(A.33)}
\end{aligned}$$

the first part is analogous to the undegenerate case Ψ'_D.

$$\begin{aligned}
\delta\Psi_\mathcal{F} &= \mu\,\mathrm{tr}(\epsilon_D(\bar{\mathcal{R}}^T\delta\mathcal{F} + \delta\mathcal{F}^T\bar{\mathcal{R}})) + \frac{\lambda}{2}\,\mathrm{tr}(\epsilon_D)\,\mathrm{tr}(\bar{\mathcal{R}}^T\delta\mathcal{F} + \delta\mathcal{F}^T\bar{\mathcal{R}}) && \text{(A.34)}\\
&= 2\mu\,\mathrm{tr}(\epsilon_D\bar{\mathcal{R}}^T\delta\mathcal{F}) + \lambda\,\mathrm{tr}(\epsilon_D)\,\mathrm{tr}(\bar{\mathcal{R}}^T\delta\mathcal{F}) && \text{(A.35)}\\
&= \bar{\mathcal{R}}\big[2\mu\epsilon_D + \lambda\,\mathrm{tr}(\epsilon_D)\mathcal{I}\big] : \delta\mathcal{F} && \text{(A.36)}\\
&= \hat{\mathcal{P}}_D : \delta\mathcal{F} && \text{(A.37)}
\end{aligned}$$

the second part does not vanish for degenerate configurations, and becomes

$$\begin{aligned}
\delta\Psi_{\bar{\mathcal{R}}} &= \mu\,\mathrm{tr}(\epsilon_D(\delta\bar{\mathcal{R}}^T\mathcal{F} + \mathcal{F}^T\delta\bar{\mathcal{R}})) + \frac{\lambda}{2}\,\mathrm{tr}(\epsilon_D)\,\mathrm{tr}(\delta\bar{\mathcal{R}}^T\mathcal{F} + \mathcal{F}^T\delta\bar{\mathcal{R}}) && \text{(A.38)}\\
&= 2\mu\,\mathrm{tr}(\epsilon_D\mathcal{F}^T\delta\bar{\mathcal{R}}) + \lambda\,\mathrm{tr}(\epsilon_D)\,\mathrm{tr}(\mathcal{F}^T\delta\bar{\mathcal{R}}) && \text{(A.39)}\\
&= \mathcal{F}\big[2\mu\epsilon_D + \lambda\,\mathrm{tr}(\epsilon_D)\mathcal{I}\big] : \delta\bar{\mathcal{R}} && \text{(A.40)}\\
&\vdots\\
&= \check{\mathcal{P}}_D : \delta\mathcal{F} && \text{(A.41)}
\end{aligned}$$

where the modified rotation differential $\delta\bar{\mathcal{R}}$ can be computed from the polar decomposition $\bar{\mathcal{F}} = \bar{\mathcal{R}}\bar{\mathcal{S}}$ using Equation (3.13), that requires the evaluation of $\delta\bar{\mathcal{F}} = \delta\mathcal{F} + \delta\bar{\mathcal{U}}\mathcal{D}_m^{-1}$. The term $\delta\bar{\mathcal{U}}$ involves the DAPD magnitudes $\hat{\boldsymbol{d}}_c$ and $\lambda(w,h)$ and their differentials, detailed in Section 4.4.2, which ultimately depend on the persistent collapse feature pair (A,B) and the instantaneous $\mathcal{F}$ and $\delta\mathcal{F}$. Unfortunately, due to this highly complex functional dependency we were unable to find a compact expression for $\check{\mathcal{P}}_D$ similar to the one obtained for $\hat{\mathcal{P}}_D$. However, we can still reason about Equation (A.40) and conclude that (i) the computational cost of evaluating $\check{\mathcal{P}}_D$ is necessarily higher than $\hat{\mathcal{P}}_D$ due to the complexity of the $\bar{\mathcal{R}}$ term and that (ii) $\check{\mathcal{P}}_D$ vanishes if we truncate the rotation differentials $\delta\bar{\mathcal{R}} = 0$.

Finally, the total energy differential

$$\begin{aligned}
\delta\Psi_D &= \delta\Psi_\mathcal{F} + \delta\Psi_{\bar{\mathcal{R}}} && \text{(A.42)}\\
&= (\hat{\mathcal{P}}_D + \check{\mathcal{P}}_D) : \delta\mathcal{F} && \text{(A.43)}
\end{aligned}$$

yields the exact stress and stress differential

$$\mathcal{P}_D = \hat{\mathcal{P}}_D + \check{\mathcal{P}}_D \tag{A.44}$$

$$\delta\mathcal{P}_D = \delta\hat{\mathcal{P}}_D + \delta\check{\mathcal{P}}_D \tag{A.45}$$

where the first summands are analogous to the undegenerate case

$$\hat{\mathcal{P}}_D = \bar{\mathcal{R}}\left[2\mu\epsilon_D + \lambda\,\mathrm{tr}(\epsilon_D)\mathcal{I}\right] \tag{A.46}$$

$$\delta\hat{\mathcal{P}}_D = 2\mu\delta\bar{\mathcal{F}} + \lambda\,\mathrm{tr}(\bar{\mathcal{R}}^T\delta\bar{\mathcal{F}})\bar{\mathcal{R}} + [\lambda\,\mathrm{tr}(\epsilon_D) - 2\mu]\delta\bar{\mathcal{R}} \tag{A.47}$$

and $\check{\mathcal{P}}_D$, $\delta\check{\mathcal{P}}_D$ have no known compact expression, but vanish simultaneously if we assume $\delta\bar{\mathcal{R}} = 0$.

A.2 Computing the potential of a vector field

Given a vector field $\vec{g}(x, y)$ we want to recover the scalar potential field $h(x, y)$ that generated it, if it exists. To do so, we will solve the partial differential equation

$$\vec{g}(x, y) = \nabla h(x, y)$$

that requires a boundary condition to yield a fully determinate solution. In our case, the boundary condition will be $h(x_0, y_0) = h_0$.

For our purposes, the input vector field $g_{i,j}$ and the solution scalar field $h_{i,j}$ are sampled on an axis-aligned regular grid $[1..N_x] \times [1..N_y]$ formed by rectangular cells with sizes $\Delta x \times \Delta y$. In this specific circumstances we can easily solve the partial differential equation at the grid cells using finite differences

$$\frac{h_{i+1,j} - h_{i,j}}{\Delta x} = \frac{\boldsymbol{f}^x_{i+1,j} + \boldsymbol{f}^x_{i,j}}{2}$$

$$\frac{h_{i,j+1} - h_{i,j}}{\Delta y} = \frac{\boldsymbol{f}^y_{i,j+1} + \boldsymbol{f}^y_{i,j}}{2}$$

Assuming that a solution exists, any discrete integration path in a conservative vector field derived from a scalar potential must yield approximately the same results. Therefore, we can (i) first propagate the boundary value $h_{i_0,j_0} = h_0$, both forward and backwards, along one axis (eg. X) using the first equation to obtain a 1-dimensional solution for $h_{1\ldots i_0\ldots N_x,j_0}$, and, (ii) afterwards use the solutions h_{i,j_0} as the boundary conditions for N_y 1-dimensional problems $h_{i,1\ldots j_0\ldots N_y}$ in the orthogonal axis Y, solved with the second equation.

If the vector field $g_{i,j}$ is actually the gradient of a scalar field, repeating the previous process using two different axis orderings, $X \to Y$ and $Y \to X$, must yield the same approximate solution. Therefore, if the scalar fields recovered for a given vector field are different we can effectively discard that the vector field is conservative. This result is used in Section 4.4.6. Figure A.1 shows the identical scalar fields recovered from an actually conservative force field.

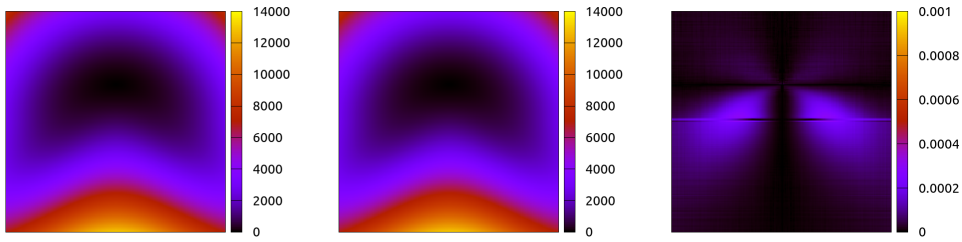

Figure A.1: Energy fields recovered from a force field using $X \to Y$ (left) and $Y \to X$ (middle) paths. Notice the negligible relative error below 0.001 (right).

A.3 Exact V-F and E-E vector area contact normals

We prove that for the basic vertex-face and edge-edge contact cases between a pair of triangulated surfaces, the vector area contact normal $\hat{\boldsymbol{n}}_V$ is exactly the same that would be computed by the proximity-based approach

- V-F: A vertex $\boldsymbol{v}_A$ with degree k on Γ_A pierces a triangular face T_B on Γ_B, resulting in a flat k-sided polygonal intersection curve completely included in T_B. The vector area returns the polygon area times the triangle unit normal, as expected.

- E-E: A pair of edges pierce each other's adjacent faces in 4 intersection points $\boldsymbol{q}_0 \dots \boldsymbol{q}_3$ that define a tetrahedron-shaped intersection volume Ω^i_{AB}. The tetrahedron is bound by 2 triangle faces on each Γ^i_A, Γ^i_B, adjacent to their contributed triangle edge segments $\boldsymbol{e}_A = \boldsymbol{q}_1 - \boldsymbol{q}_0$ and $\boldsymbol{e}_B = \boldsymbol{q}_3 - \boldsymbol{q}_2$. The proximity-based contact normal direction would be aligned with $\boldsymbol{e}_B \times \boldsymbol{e}_A$. The vector area computed on Γ^i_A as the sum the area weighted triangle normals of its contributed faces T_{012} and T_{031} adjacent to $\boldsymbol{e}_A$ yields the same result

$$
\begin{aligned}
\boldsymbol{N}_V &= (\boldsymbol{q}_1 - \boldsymbol{q}_0) \times (\boldsymbol{q}_2 - \boldsymbol{q}_0) + (\boldsymbol{q}_3 - \boldsymbol{q}_0) \times (\boldsymbol{q}_1 - \boldsymbol{q}_0) \\
&= (\boldsymbol{q}_1 - \boldsymbol{q}_0) \times (\boldsymbol{q}_2 - \boldsymbol{q}_0) - (\boldsymbol{q}_1 - \boldsymbol{q}_0) \times (\boldsymbol{q}_3 - \boldsymbol{q}_0) \\
&= (\boldsymbol{q}_1 - \boldsymbol{q}_0) \times \big((\boldsymbol{q}_2 - \boldsymbol{q}_0) - (\boldsymbol{q}_3 - \boldsymbol{q}_0)\big) \\
&= (\boldsymbol{q}_1 - \boldsymbol{q}_0) \times (\boldsymbol{q}_2 - \boldsymbol{q}_3) \\
&= \boldsymbol{e}_A \times -\boldsymbol{e}_B \\
&= \boldsymbol{e}_B \times \boldsymbol{e}_A
\end{aligned}
$$

A.4 Vector area transformation

Given the affine deformation $\mathcal{F} = \mathcal{D}_s \mathcal{D}_m^{-1}$ that relates reference $\mathcal{D}_m$ and deformed $\mathcal{D}_s$ configurations of a simplex in 2D or 3D, we show how to compute the required

summation over all triangles of an embedded geometry patch from its precomputed vector area in reference configuration in $\mathcal{O}(1)$, regardless of the patch triangle count. The undeformed vector area is:

$$\boldsymbol{N}_V^m = \frac{1}{2}\sum(\boldsymbol{u} \times \boldsymbol{v})$$

When the patch deforms, we use the cross-product affine transformation $\mathcal{F}\boldsymbol{u} \times \mathcal{F}\boldsymbol{v} = \det(\mathcal{F})\mathcal{F}^{-T}(\boldsymbol{u}\times\boldsymbol{v})$ from [Bar84], where $\det(\mathcal{F})\mathcal{F}^{-T}$ is the transposed *adjugate* $\operatorname{adj}(\mathcal{F})^T$ to obtain:

$$\boldsymbol{N}_V^s = \frac{1}{2}\sum \operatorname{adj}(\mathcal{F})^T(\boldsymbol{u} \times \boldsymbol{v}) = \frac{1}{2}\operatorname{adj}(\mathcal{F})^T\boldsymbol{N}_V^m$$

that can be computed in $\mathcal{O}(1)$ if $\boldsymbol{N}_V^m$ is available.

Bibliography

[AB03] Uri M Ascher and Eddy Boxerman. On the modified conjugate gradient method in cloth simulation. *The Visual Computer*, 19(7-8):526–531, 2003.

[ACPR94] Uri M. Ascher, Hongsheng Chin, Linda R. Petzold, and Sebastian Reich. Stabilization of constrained mechanical systems with daes and invariant manifolds. *J. Mech. Struct. Machines*, 23:135–157, 1994.

[ACSYD05] Pierre Alliez, David Cohen-Steiner, Mariette Yvinec, and Mathieu Desbrun. Variational tetrahedral meshing. *ACM Transactions on Graphics (TOG)*, 24(3):617–625, 2005.

[AFC+10] Jérémie Allard, François Faure, Hadrien Courtecuisse, Florent Falipou, Christian Duriez, and Paul G Kry. Volume contact constraints at arbitrary resolution. In *ACM Transactions on Graphics (TOG)*, volume 29, page 82. ACM, 2010.

[AP97] M. Anitescu and F. A. Potra. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *NONLINEAR DYNAMICS*, 14:231–247, 1997.

[Bar84] Alan H. Barr. Global and local deformations of solid primitives. *SIGGRAPH Comput. Graph.*, 18(3):21–30, January 1984.

[Bar96] David Baraff. Linear-time dynamics using lagrange multipliers. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 137–146. ACM, 1996.

[Bar12] Jernej Barbic. Exact corotational linear fem stiffness matrix. Technical report, Citeseer, 2012.

[BC14] Adam W Bargteil and Elaine Cohen. Animation of deformable bodies with quadratic bézier finite elements. *ACM Transactions on Graphics (TOG)*, 33(3):27, 2014.

[BET14] Jan Bender, Kenny Erleben, and Jeff Trinkle. Interactive simulation of rigid body dynamics in computer graphics. In *Computer Graphics Forum*, volume 33, pages 246–270. Wiley Online Library, 2014.

[BFA02] Robert Bridson, Ronald Fedkiw, and John Anderson. Robust treatment of collisions, contact and friction for cloth animation. In *ACM Transactions on Graphics (ToG)*, volume 21, pages 594–603. ACM, 2002.

[BKCW14] Jan Bender, Dan Koschier, Patrick Charrier, and Daniel Weber. Position-based simulation of continuous materials. *Computers & Graphics*, 44:1–10, 2014.

[BML+14] Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. Projective dynamics: fusing constraint projections for fast simulation. *ACM Transactions on Graphics (TOG)*, 33(4):154, 2014.

[BMO+14] Jan Bender, Matthias Müller, Miguel A. Otaduy, Matthias Teschner, and Miles Macklin. A survey on position-based simulation methods in computer graphics. *Computer Graphics Forum*, 33(6):228–251, 2014.

[BMOT13] Jan Bender, Matthias Müller, Miguel A Otaduy, and Matthias Teschner. Position-based methods for the simulation of solid objects in computer graphics. *EUROGRAPHICS 2013 State of the Art Reports*, 2013.

[BPWG07] Mario Botsch, Mark Pauly, Martin Wicke, and Markus Gross. Adaptive space deformations based on rigid cells. In *Computer Graphics Forum*, volume 26, pages 339–347. Wiley Online Library, 2007.

[BW98] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 43–54, New York, NY, USA, 1998. ACM.

[BWK03] David Baraff, Andrew Witkin, and Michael Kass. Untangling cloth. *ACM Trans. Graph.*, 22(3):862–870, July 2003.

[CdGDS13] Keenan Crane, Fernando de Goes, Mathieu Desbrun, and Peter Schröder. Digital geometry processing with discrete exterior calculus. In *ACM SIGGRAPH 2013 Courses*, SIGGRAPH '13, pages 7:1–7:126, New York, NY, USA, 2013. ACM.

[CFS14] O. Civit-Flores and A. Susín. Robust treatment of degenerate elements in interactive corotational fem simulations. *Computer Graphics Forum*, 33(6):298–309, 2014.

[CMS01] Hamish Carr, Torsten Möller, and Jack Snoeyink. Simplicial subdivisions and sampling artifacts. In *Proceedings of the conference on Visualization'01*, pages 99–106. IEEE Computer Society, 2001.

[CP03] Michael B Cline and Dinesh K Pai. Post-stabilization for rigid body simulation with contact and constraints. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 3, pages 3744–3751. IEEE, 2003.

[CPSS10] Isaac Chao, Ulrich Pinkall, Patrick Sanan, and Peter Schröder. A simple geometric model for elastic deformations. *ACM Trans. Graph.*, 29(4):38:1–38:6, July 2010.

[CSRL01] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[DAK04] Christian Duriez, Claude Andriot, and Abderrahmane Kheddar. Signorini's contact model for deformable objects in haptic simulations. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 4, pages 3232–3237. IEEE, 2004.

[DDKA06] Christian Duriez, Frederic Dubois, Abderrahmane Kheddar, and Claude Andriot. Realistic haptic rendering of interacting deformable objects in virtual environments. *Visualization and Computer Graphics, IEEE Transactions on*, 12(1):36–47, 2006.

[EBM12] Martin Eisemann, Pablo Bauszat, and Marcus Magnor. Implicit object space partitioning: The no-memory BVH. Technical Report 16, Computer Graphics Lab, TU Braunschweig, January 2012.

[EGH00] Robert Eymard, Thierry Gallouët, and Raphaèle Herbin. Finite volume methods. *Handbook of numerical analysis*, 7:713–1018, 2000.

[ES04] D.H. Eberly and K. Shoemake. *Game Physics*. Interactive 3D technology series. Taylor & Francis, 2004.

[ESHD05] Kenny Erleben, Jon Sporring, Knud Henriksen, and Kenrik Dohlman. *Physics-based Animation (Graphics Series)*. Charles River Media, Inc., Rockland, MA, USA, 2005.

[Fel04] Carlos A Felippa. Introduction to finite element methods. *Course Notes, Department of Aerospace Engineeing Sciences, University of Colorado at Boulder*, 2004.

[FL01] Susan Fisher and Ming C Lin. *Deformed distance fields for simulation of non-penetrating flexible bodies.* Springer, 2001.

[FM04] Thomas-Peter Fries and Hermann G. Matthies. Classification and overview of meshfree methods. Technical report, Institute of Scientific Computing Technical University Braunschweig Brunswick, Germany, 2004.

[GFT08] Marc Gissler, Udo Frese, and Matthias Teschner. Exact distance computation for deformable objects. *Proc. Computer Animation and Social Agents CASA*, 8:47–54, 2008.

[GPHW05] R Brent Gillespie, Volkan Patoglu, Islam I Hussein, and ER Westervelt. On-line symbolic constraint embedding for simulation of hybrid dynamical systems. *Multibody System Dynamics*, 14(3-4):387–417, 2005.

[GPS02] H. Goldstein, C.P. Poole, and J.L. Safko. *Classical Mechanics.* Addison Wesley, 2002.

[Hau04] Michael Hauth. *Visual simulation of deformable models.* PhD thesis, Universität Tübingen, 2004.

[HET01] Michael Hauth, Olaf Etzmuss, and Universität Tübingen. A high performance solver for the animation of deformable objects using advanced numerical methods. In *In Proc. Eurographics 2001 (2001), Chalmers A., Rhyne T.-M., (Eds*, pages 319–328, 2001.

[HFR08] Everton Hermann, François Faure, and Bruno Raffin. Ray-traced collision detection for deformable bodies. In *GRAPP 2008-3rd International Conference on Computer Graphics Theory and Applications*, pages 293–299. INSTICC, 2008.

[HS04] Michael Hauth and Wolfgang Strasser. Corotational simulation of deformable solids. *Journal of WSCG*, 12(1-3):2–9, 2004.

[HTK+04] Bruno Heidelberger, Matthias Teschner, Richard Keiser, Matthias Müller, , and Markus Gross. Consistent penetration depth estimation for deformable collision response. In *Vision, Modeling, and Visualization 2004: Proceedings, November 16-18, 2004, Standford, USA*, page 339. IOS Press, 2004.

[ITF04] G. Irving, J. Teran, and R. Fedkiw. Invertible finite elements for robust simulation of large deformation. In *Symposium on Computer Animation*, pages 131–140, Aire-la-Ville, Switzerland, 2004.

[JP04] Doug L. James and Dinesh K. Pai. Bd-tree: Output-sensitive collision detection for reduced deformable models. *ACM Trans. Graph.*, 23(3):393–398, August 2004.

[Kel95] C.T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*. Frontiers in Applied Mathematics. Society for Industrial and Applied Mathematics, 1995.

[KHM+98] James T. Klosowski, Martin Held, Joseph S. B. Mitchell, Henry Sowizral, and Karel Zikan. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS*, 4:21–36, 1998.

[KMBG09] Peter Kaufmann, Sebastian Martin, Mario Botsch, and Markus Gross. Flexible simulation of deformable models using discontinuous galerkin fem. *Graphical Models*, 71(4):153–167, 2009.

[KNF04] Stephan Kimmerle, Matthieu Nesme, and François Faure. Hierarchy accelerated stochastic collision detection. In *9th International Workshop on Vision, Modeling, and Visualization, VMV 2004*, 2004.

[Lac07] Claude Lacoursière. *Ghosts and machines: regularized variational methods for interactive simulations of multibodies with dry frictional contacts*. PhD thesis, 2007.

[LS07] François Labelle and Jonathan Richard Shewchuk. Isosurface stuffing: Fast tetrahedral meshes with good dihedral angles. *ACM Trans. Graph.*, 26(3), July 2007.

[MCKM14] Matthias Müller, Nuttapong Chentanez, Tae-Yong Kim, and Miles Macklin. Strain based dynamics. In *Proceedings of ACM SIGGRAPH/EUROGRAPHICS Symposium on Computer Animation (SCA)(Copenhagen*, volume 2, 2014.

[MCKM15] Matthias Müller, Nuttapong Chentanez, Tae-Yong Kim, and Miles Macklin. Air meshes for robust collision handling. *ACM Trans. Graph.*, 34(4):133:1–133:9, July 2015.

[MG04] Matthias Müller and Markus Gross. Interactive virtual materials. In *Proceedings of Graphics Interface 2004*, pages 239–246, Waterloo, Ontario, Canada, 2004.

[MHHR07] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118, 2007.

[MZS+11a] Aleka McAdams, Yongning Zhu, Andrew Selle, Mark Empey, Rasmus Tamstorf, Joseph Teran, and Eftychios Sifakis. Efficient elasticity for character skinning with contact and collisions. In *ACM SIGGRAPH 2011*, pages 37:1–37:12, New York, USA, 2011.

[MZS+11b] Aleka McAdams, Yongning Zhu, Andrew Selle, Mark Empey, Rasmus Tamstorf, Joseph Teran, and Eftychios Sifakis. Efficient elasticity for character skinning with contact and collisions. *ACM Trans. Graph.*, 30(4):37:1–37:12, July 2011.

[NKJF09] Matthieu Nesme, Paul G Kry, Lenka Jeřábková, and François Faure. Preserving topology and elasticity for embedded deformable models. *ACM Transactions on Graphics (TOG)*, 28(3):52, 2009.

[NMK+06] Andrew Nealen, Matthias Müller, Richard Keiser, Eddy Boxerman, and Mark Carlson. Physically based deformable models in computer graphics. In *Computer Graphics Forum*, volume 25, pages 809–836. Wiley Online Library, 2006.

[NPF05] Matthieu Nesme, Yohan Payan, and François Faure. Efficient, Physically Plausible Finite Elements. In *Eurographics*, Dublin, Ireland, 2005.

[OGRG07] Miguel A. Otaduy, Daniel Germann, Stephane Redon, and Markus Gross. Adaptive deformations with fast tight bounds. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '07, pages 181–190, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[OTSG09] Miguel A Otaduy, Rasmus Tamstorf, Denis Steinemann, and Markus Gross. Implicit contact handling for deformable objects. In *Computer Graphics Forum*, volume 28, pages 559–568. Wiley Online Library, 2009.

[PO09] Eric G. Parker and James F. O'Brien. Real-time deformation and fracture in a game environment. In *Symposium on Computer Animation*, pages 165–175, New York, NY, USA, 2009.

[Rot02] Samuel Hans Martin Roth. *Bernstein–Bézier Representations for Facial Surgery Simulation.* PhD thesis, ETH, Zurich, 2002.

[SB12] Eftychios Sifakis and Jernej Barbic. Fem simulation of 3d deformable solids: A practitioner's guide to theory, discretization and model reduction. In *ACM SIGGRAPH 2012 Courses*, SIGGRAPH '12, pages 20:1–20:50, New York, NY, USA, 2012. ACM.

[SD92] Ken Shoemake and Tom Duff. Matrix animation and polar decomposition. In *Proceedings of Graphics Interface '92*, pages 258–264, San Francisco, CA, USA, 1992.

[SGO09] Sara C. Schvartzman, Jorge Gascón, and Miguel A. Otaduy. Bounded normal trees for reduced deformations of triangulated surfaces. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '09, pages 75–82, New York, NY, USA, 2009. ACM.

[SHST12] Alexey Stomakhin, Russell Howes, Craig Schroeder, and Joseph M. Teran. Energetically consistent invertible elasticity. In *Symposium on Computer Animation*, pages 25–32, Aire-la-Ville, Switzerland, 2012.

[SKPSH13] Christian Schüller, Ladislav Kavan, Daniele Panozzo, and Olga Sorkine-Hornung. Locally injective mappings. In *Computer Graphics Forum*, volume 32, pages 125–135. Wiley Online Library, 2013.

[SLJB13] David A. Stuart, Joshua A. Levine, Ben Jones, and Adam W. Bargteil. Automatic construction of coarse, high-quality tetrahedralizations that enclose and approximate surfaces for animation. In *Proceedings of Motion on Games*, MIG '13, pages 191:213–191:222, New York, NY, USA, 2013. ACM.

[SLM06] Martin Servin, Claude Lacoursiere, and Niklas Melin. Interactive simulation of elastic deformable materials. In *Proceedings of SIGRAD Conference*, pages 22–32, 2006.

[SMT08] Eftychios Sifakis, Sebastian Marino, and Joseph Teran. Globally coupled collision handling using volume preserving impulses. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 147–153. Eurographics Association, 2008.

[SS86] Youcef Saad and Martin H Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7(3):856–869, July 1986.

[SSIF07] Eftychios Sifakis, Tamar Shinar, Geoffrey Irving, and Ronald Fedkiw. Hybrid simulation of deformable solids. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 81–90. Eurographics Association, 2007.

[ST08] Ruediger Schmedding and Matthias Teschner. Inversion handling for stable deformable modeling. *The Visual Computer*, 24(7):625–633, 2008.

[TBHF03] J. Teran, S. Blemker, V. Ng Thow Hing, and R. Fedkiw. Finite volume methods for the simulation of skeletal muscle. In *Symposium on Computer Animation*, pages 68–74, Aire-la-Ville, Switzerland, 2003.

[TKH+05] Matthias Teschner, Stefan Kimmerle, Bruno Heidelberger, Gabriel Zachmann, Laks Raghupathi, Arnulph Fuhrmann, M-P Cani, François Faure, Nadia Magnenat-Thalmann, Wolfgang Strasser, et al. Collision detection for deformable objects. In *Computer graphics forum*, volume 24, pages 61–81. Wiley Online Library, 2005.

[TMOT12] Min Tang, Dinesh Manocha, Miguel A. Otaduy, and Ruofeng Tong. Continuous penalty forces. *ACM Trans. Graph.*, 31(4):107:1–107:9, July 2012.

[TMY+11] Min Tang, Dinesh Manocha, Sung-Eui Yoon, Peng Du, Jae-Pil Heo, and Ruo-Feng Tong. Volccd: Fast continuous collision culling between deforming volume meshes. *ACM Trans. Graph.*, 30(5):111:1–111:15, October 2011.

[TNGF15] Maxime Tournier, Matthieu Nesme, Benjamin Gilles, and François Faure. Stable constrained dynamics. *ACM transactions on Graphics, Proceedings of ACM SIGGRAPH*, page 0, 2015.

[TPBF87] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *ACM Siggraph Computer Graphics*, volume 21, pages 205–214. ACM, 1987.

[TSIF05] Joseph Teran, Eftychios Sifakis, Geoffrey Irving, and Ronald Fedkiw. Robust quasistatic finite elements and flesh simulation. In *Symposium on Computer Animation*, pages 181–190, New York, NY, USA, 2005.

[WGW90] Andrew Witkin, Michael Gleicher, and William Welch. Interactive dynamics. *SIGGRAPH Comput. Graph.*, 24(2):11–21, February 1990.

[WLG06] Martin Wicke, Hermes Lanker, and Markus Gross. Untangling cloth with boundaries. *Proc. of Vision, Modeling, and Visualization*, pages 349–356, 2006.

[Wri06] Peter Wriggers. *Computational contact mechanics*. Springer, Berlin, New York, 2006.

[WT08a] Chris Wojtan and Greg Turk. Fast viscoelastic behavior with thin features. *ACM Trans. Graph.*, 27(3):47:1–47:8, August 2008.

[WT08b] Chris Wojtan and Greg Turk. Fast viscoelastic behavior with thin features. In *ACM transactions on graphics (TOG)*, volume 27, page 47. ACM, 2008.